UNISYS

A Series
DMS II
Inquiry
Operations Guide

Relative to Release
Level Mark 3.7.0

Priced Item

July 1987
Distribution Code SE
Printed in U S America
1164035.370

UNISYS

UNISYS **Publication Change Notice (PCN)**

Date	Form—PCN number
May 28, 1989	1164035-003

Title
A Series DMSII Inquiry Operations Guide
(Relative to the Mark 3.8.0 System Software Release)

Description

This letter announces the availability of PCN-003 to the A Series DMSII Inquiry Operations Guide, form number 1164035, dated December 1985. This PCN contains revisions and additions relative to the Mark 3.8.0 System Software Release.

Changes to the text are indicated by vertical black lines on the affected pages.

Replace These Pages	Add These Pages
211 through 221	iii
	8A

Retain the PCN cover sheet as a record of changes made to the basic publication.

Copyright (c) 1989 Unisys Corporation
Unisys Corporation
All Rights Reserved

Printed in U S America

UNISYS **Publication Change Notice (PCN)**

Date: July 24, 1987

Form—PCN number: 1164035-002

Title: A Series DMSII Inquiry Operations Guide
(Relative to the Mark 3.7 System Software Release)

Description

This letter announces the availability of PCN-002 to the A Series DMSII Inquiry Operations Guide form number 1164035, dated July 1987. This PCN includes revisions and additions relative to the Mark 3.7 System Software Release. Changes to the text are indicated by vertical black lines on the affected pages.

Replace These Pages	Insert These Pages
Title	204A
TOC iii	
1	
3	
25	
93	
95	
203	
211	
217	
221	

Retain this PCN cover sheet as a record of changes made to the basic publication.

Copies of this PCN can be ordered from the Publications Distribution Center, Plymouth, Michigan.

COPYRIGHT © 1987
Unisys Corporation

UNISYS Publication Change Notice (PCN)

Date	Form--PCN number
10-25-85	1164035-001

Title

A Series DMSII Inquiry Software Operation Guide PCN-001

Description

This PCN includes additions and revisions made to the manual to reflect the Mark 3.6 Software Release. Changes to the text are indicated by a vertical black bar on the affected pages.

Replace these pages	Add these pages
title	
Table of Contents	128A
3	182A
127	
181	
183	
191	
215	
219	
225	
227	

Retain this PCN cover sheet as a record of changes made to the basic publication.

COPYRIGHT © 1984, 1985
Unisys Corporation

UNISYS

A Series
DMS II
Inquiry
Operations Guide

Copyright© 1987, Unisys Corporation
All Rights Reserved
Unisys is a trademark of Unisys Corporation
Previous Title: DMSII Inquiry Software Operation Guide

Relative to Release
Level Mark 3.7.0

Priced Item

July 1987
Distribution Code SE
Printed in U S America
1164035.370

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual living or otherwise, or that of any group or association, is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication may be forwarded using the Remarks form at the back of the manual; submitted on a Field Communication Form with the Class specified as 2 (System Software), the Type specified as 1 (F.T.R.), and the Product specified as the form number of this manual (1164035.370); or addressed directly to Unisys Corporation, Software Documentation, 19 Morgan, Irvine, California, 92718, U.S.A.

PCN 1164035-002 Release 3.7 July 1987

Page Status

Page	Issue
iii	PCN-003
iv	Blank
TOC	PCN-002
1 through 3	PCN-002
4	Blank
5 through 8	Original
8A	PCN-003
8B	Blank
9 through 24	Original
25 through 26	PCN-002
27 through 47	Original
48	Blank
49 through 92	Original
93 through 96	PCN-002
97 through 126	Original
127 through 128A	PCN-001
128B	Blank
129 through 130	PCN-001
131 through 180	Original
181 through 182A	PCN-001
182B	Blank
183 through 184	PCN-001
185 through 190	Original
191 through 192	PCN-001
193 through 202	Original
203 through 204B	PCN-002
205 through 209	Original
210	Blank
211 through 222	PCN-003
223 through 224	Original
225 through 228	PCN-001

CONTENTS

1 INTRODUCTION 1
SCOPE OF THIS MANUAL 1
ORGANIZATION OF THIS MANUAL. 2
RELATED DOCUMENTS. 3

2 USE OF INQUIRY 5
SAMPLE DATA BASE 5
BASIC INQUIRY PROCESS. 7
Record Selection 8
Item And Value Display 12
Combining Record Selection And Item Display. 13
EMBEDDED STRUCTURES. 15
USE OF FUNCTIONS 17
Boolean Functions. 17
Arithmetic Functions 18
VIRTUAL ITEMS. 21
UNDEFINED ITEMS. 23
RELATED RECORDS ACROSS DATA SETS 24
DATA BASE MODIFICATION 26
Update 26
Create And Delete. 28
Modification Of Audited Data Bases 31
OUTPUT CONTROL AND FORMATTING. 32
DISPLAY Command. 32
Embedded Structures. 33
Formatting Attributes. 35
INQUIRY REPORT 41
INQUIRY EXTRACT. 46

3 INQUIRY COMMANDS 49
INTRODUCTION 49
SUMMARY OF INQUIRY COMMANDS. 49
COMMON SYNTACTIC ITEMS 53
<selection condition>. 53
<display list> 59
<sort option>. 62
Functions. 66
DATA SET FUNCTIONS 69
CURRENT RECORD FUNCTIONS 73
INQUIRY COMMANDS 76
SELECT Command 76
DISPLAY Command. 80
SELECT/DISPLAY Command 88
ATTACH Command 91
CLEAR Command. 93
CREATE Command 97
DEFINE Command 101
DELETE Command 104
DETACH Command 106

EDIT Command 108
EXTRACT Command. 112
FILE Command 121
GENERATE Command 125
HELP Command 132
NEXT Command 134
OPTIONS Command. 136
PRINTER Command. 140
QUIT Command 144
RECALL Command 145
REDEFINE Command 148
REPEAT Command 150
REPORT Command 153
RESTORE Command. 163
SAVE Command 166
SET Command. 169
SHOW Command 173
SORT Command 178
SUMMARY Command. 180
TERMINAL Command 184
TITLE Command. 187
UPDATE Command 192
VIRTUAL Command. 197

A HOW TO READ RAILROAD DIAGRAMS. 199

B GLOSSARY 203

C GENERATING AND RUNNING INQUIRY 211
INQUIRY FILES. 212
INQUIRY GENERATION 213
Generation From Remote Terminals 213
Remote Generation Using The Nozip Option 217
Generation From Job Decks. 218
INITIATING AND RUNNING INQUIRY 221
Required Files 221
Efficiency Considerations. 221
Run-time Diagnostic Options. 222
Operational Description. 222
Running Inquiry From Terminals 222

INDEX 225

1 INTRODUCTION

The DMSII Inquiry program provides an on-line, interactive system that can be used to examine or modify information contained in a DMSII data base. Inquiry contains a small number of input statements, which can be combined to perform complex operations. Thus, the design of Inquiry allows data base users who are relatively unfamiliar with data base design concepts to effectively access and use the information contained in DMSII data bases.

Inquiry provides the following capabilities:

1. Examination of data base information. Inquiry users may examine either the contents of a data base or the data base description from remote terminals. Inquiry is able to access information in any part of a data base, regardless of the complexity of the data base, and is always able to produce the requested information, even if linear file searches must be performed. Whenever possible, Inquiry takes advantage of sets when accessing data base information.

2. Database modification. Inquiry users may use input statements to modify, create, or delete records within data bases. (Data base administrators may use DASDL remaps and logical data bases to define which parts of a data base allow Inquiry access and which users may access these parts.)

3. Report generation. The Inquiry REPORT facility allows simple reports to be generated from data base information.

4. Extraction of data base information. The Inquiry EXTRACT facility allows selected data base information to be extracted to a standard data file.

SCOPE OF THIS MANUAL

This manual is designed to provide both a description of the operations required to use the Inquiry program and formal syntactic and semantic descriptions of statements used as input to the program. The reader is assumed to have a basic understanding of the concepts and terminology associated with the Unisys Data and Structure Definition Language (DASDL), including items, records, sets, data sets, embedded sets, disjoint sets, and keys.

ORGANIZATION OF THIS MANUAL

The manual is organized as follows.

Section 1 Introduction

This section presents an introduction to the DMSII Inquiry manual.

Section 2 Use of Inquiry

This section contains illustrations, including detailed examples, of the operations involved in selecting, displaying, and modifying data base records using Inquiry. Descriptions of the Inquiry REPORT and EXTRACT facilities are also included.

Section 3 Inquiry Commands

This section contains syntax and semantics for Inquiry input commands.

Appendix A How to Read Railroad Diagrams

This section describes how to read railroad diagrams.

Appendix B Glossary

This section defines the terms used in DMSII literature and the Inquiry manual.

Appendix C Generating and Running Inquiry

This section describes the files required to generate the Inquiry program and lists the necessary steps for generating and initiating the program.

RELATED DOCUMENTS

The following A Series documents contain information related to the material in this manual:

Document	Form No.
DMSII Data and Structure Definition Language (DASDL) Programming Reference Manual formerly DMSII Data and Structure Definition Language (DASDL) Reference Manual	1163805
DMSII User Language Interface Programming Guide formerly DMSII User Language Interface Software Operation Guide	1180536

2. USE OF INQUIRY

This section illustrates the operations involved in the selection, display and modification of data base records using Inquiry and in the use of the Inquiry REPORT and EXTRACT facilities.

For all examples in the following two sections that display both user input and Inquiry output, user input is indicated by an arrow (->). In addition, output examples are displayed for a terminal width of 70 characters.

SAMPLE DATA BASE

The following DASDL input creates the data base that is used for the examples in the remainder of this manual:

```
$SET LIST ALLINFO
OPTIONS(AUDIT);
DEFAULTS (KIND=PACK,PACKNAME=SOFTDOC);
RSTART RESTART DATA SET
 (ALPHAID ALPHA(7));
AUDIT TRAIL ATTRIBUTES(BLOCKSIZE = 10 SEGMENTS, PACK=SOFTDOC);
%********** GLOBAL RECORD **********
FISCAL-YEAR "NEEDS UPDATING EVERY YEAR" NUMBER (6);
POP-EMP POPULATION (50) OF EMP;
DIV-NUMBER NUMBER (5);

%****************** EMP ******************************************
EMP DATA SET  "MASTER EMPLOYEE FILE"
POPULATION = 50
(
EMP-NO                  NUMBER      (4); %ACTUAL EMP NUMBER
EMP-NAME                GROUP
    (
    EMP-LNAME              ALPHA      (10);
    EMP-FNAME              ALPHA      (10);
    );
EMP-SSN                 NUMBER      (9);
EMP-DATE-HIRED "YYMMDD" NUMBER      (6);
EMP-SEC-CLEAR "T=YES"   BOOLEAN       ;
EMP-SALARY "MONTHLY"    NUMBER    (6,2);
          %**** EMP-ED ******
EMP-ED DATA SET "EDUCATIONAL RECORDS FOR EACH EMPLOYEE"
    POPULATION = 10
    (
    ED-DATE "YYMMDD"        NUMBER      (6);
    ED-INST-NAME            ALPHA      (30);
    ED-INST-CODE            NUMBER      (4); %UNIQUE TO SCHOOL
    ED-COURSE               ALPHA      (10);
```

```
    ED-GPA                  NUMBER    (3,2); %GRADE POINT AVERAGE
    );
ED-DATE-SET             SET OF EMP-ED KEY ED-DATE I-S;
EMP-JOB-REF             REFERENCE TO JOB COUNTED;
);
EMP-NO-SET              SET OF EMP KEY EMP-NO I-S;
EMP-LNAME-SET           SET OF EMP KEY EMP-LNAME DUPLICATES I-S;
EMP-OVER10-SUBSET "VESTED EMPLOYEES" SUBSET OF EMP
                          WHERE (EMP-DATE-HIRED < 700000)
                          BIT VECTOR;

%****************** JOB ****************************************
JOB DATA SET "REC TYPE 1 IS CLERICAL, REC TYPE 2 IS NON-CLERICAL"
POPULATION = 10
(
JOB-COUNT               COUNT      (10);
JOB-TYPE                TYPE        (2);
JOB-CODE                NUMBER      (4);
JOB-TITLE               ALPHA      (30);
JOB-EMP-SUBSET       SUBSET OF EMP KEY EMP-NO;
),
  1:
(
TYPE-SPEED              NUMBER      (2);
SHORTHAND-SPEED         NUMBER      (2);
CRTUSE-REQD             BOOLEAN        ;
),
  2:
(
SPANISH-REQD            BOOLEAN        ;
DRIVING-REQD            BOOLEAN        ;
JOB-COMMENT             ALPHA      (50);
);
JOB-CODE-SET            SET OF JOB KEY JOB-CODE I-S;
JOB-MNG-SUBSET "MANAGERS" SUBSET OF JOB WHERE
                          (JOB-CODE > 4999);

%****************** ADR ****************************************
ADR DATA SET "MASTER ADDRESS FILE"
POPULATION = 50
(
ADR-EMP-NO              NUMBER      (4);
ADR-INFO                GROUP
    (
    ADR-STREET-NO           ALPHA      (30);
    ADR-ZIP                 NUMBER      (5);
    );
);
ADR-ZIP-SET             SET OF ADR KEY ADR-ZIP I-S DUPLICATES;
```

BASIC INQUIRY PROCESS

The following paragraphs illustrate basic Inquiry record-selection and item-display operations.

The Inquiry program may be used to select data base records and to display items within those records. The desired records must be present within a designated structure or must satisfy conditions, known as "selection conditions," specified in Inquiry input commands.

If Inquiry fails to find a record that satisfies the user's specification, the following message is displayed:

```
#NONE
```

If Inquiry finds a record that satisfies the user's specification, a pound sign (#) is displayed. If the user then wishes to display the selected record, or items within the selected record, the DISPLAY command is entered. If the user wishes to locate the next record that satisfies the specification, the NEXT command is entered; then, if another record satisfies the specification, Inquiry again displays a pound sign and awaits further input. This process continues until no more records meet the user's specification. At that point, Inquiry displays the following message:

```
#NO MORE
```

Examples:

In the following example, Inquiry is asked to select the first record in the data set EMP that applies to an employee hired after December 25, 1955:

```
-> SELECT EMP AT EMP-DATE-HIRED > 551225

   #
```

Next, Inquiry is asked to display the date the employee was hired, the employee-number of the employee, and the last name of the employee:

```
-> DISPLAY EMP-DATE-HIRED, EMP-NO, EMP-LNAME

   EMP-DATE-HIRED  EMP-NO  EMP-LNAME
   --------------  ------  ----------
          720312       1  FIELDS
   #
```

Each succeeding record that meets the condition specified may then be selected in turn, and the specified information may be displayed until no additional records in EMP meet the specified condition, by using the NEXT and REPEAT commands as follows:

```
-> NEXT

   #
```

In this context, the NEXT command instructs Inquiry to select the next record that meets the specified condition. The REPEAT command can then be used to display the desired information from the newly selected record without having to re-enter the DISPLAY command:

```
-> REPEAT

   EMP-DATE-HIRED  EMP-NO  EMP-LNAME
   --------------  ------  ----------
          701214       2  HARDWICK
   #
   .
   .
   .
-> NEXT

   #NO MORE
   #
```

The NO MORE display indicates that no records remain that meet the specified condition.

Record Selection

The user selects records using Inquiry by specifying selection conditions involving data base records and items, or by designating an access structure.

Use of Inquiry

If any part of the selection condition contains an item declared as a key of a set, then Inquiry attempts to use that set to find the requested records. However, Inquiry cannot use the set if it has a group item key that is not the only key of the set. In such a case, Inquiry searches the data set for the requested records.

Selection Conditions

Inquiry locates all records that satisfy the user-specified selection condition. Selection conditions are defined under <selection condition> in COMMON SYNTACTIC ITEMS in Section 3. A simple selection condition has the following form:

<item> <relational operator> <value>

In this form, <value> may either be a numeric value or the result of an arithmetic expression. Relational operators and arithmetic expressions are defined under <selection condition> in "Common Syntactic Items" in Section 3.

The logical operators AND, OR, and NOT can be used to form more complex selection conditions.

Examples:

The following SELECT command selects records in the data set ADR for addresses containing zip codes that are not equal to 50007:

```
SELECT ADR-ZIP NEQ 50007
```

The following SELECT command selects records in the data set EMP for employees hired before 1965 and who do not earn more than $3000 per month:

```
SELECT EMP-DATE-HIRED < 650101 AND NOT(EMP-SALARY > 3000)
```

The following SELECT command selects records in the embedded data set EMP-ED for employees who attended Harvard or whose grade-point averages were greater than 3.5:

```
SELECT ED-INST-NAME = "HARVARD" OR ED-GPA > 3.5
```

In the preceding example, a record must already have been selected from the master data set EMP (refer to EMBEDDED STRUCTURES in this section); if not, Inquiry displays the following message:

```
#OWNER STRUCTURE NOT YET SELECTED
```

Structure Designation

A number of sets may be associated with each data set. Unless a particular structure is designated, Inquiry attempts to locate records using these sets. Because subsets may not locate all records contained in the data set, subsets are not used unless they are specifically designated in the user's request.

The user can force Inquiry to locate records using a particular data set, set, or subset by specifying the name of the structure in the SELECT command. This procedure is known as "structure designation." If no sort specification is given, all records in the data set are selected in the order in which they appear in the designated access structure.

Examples:

The following SELECT command allows Inquiry to choose the best access method possible to select records for employees in EMP with a last name of "JONSON":

```
SELECT EMP-LNAME = "JONSON"
```

The following SELECT command designates the subset EMP-OVER10-SUBSET and causes Inquiry to use this subset to select records for EMP with a last name of "JONSON". Thus, only records from the subset that meet the selection condition are selected. That is, only employees who were hired at least 10 years previously and who are named "JONSON" are located:

```
SELECT EMP-OVER10-SUBSET AT EMP-LNAME = "JONSON"
```

The following SELECT command does not specify a set or subset to be used to select the specified record. Because no set specifies EMP-DATE-HIRED as a key, a linear search of the data set EMP is performed:

```
SELECT EMP-DATE-HIRED < 650101
```

Because the following command specifies the data set EMP, it causes Inquiry to perform a linear search of the data set EMP, even though a set could have been used to locate the record more efficiently:

```
SELECT EMP AT EMP-NO = 8
```

Qualification

If two data sets in a data base contain items with the same name, Inquiry does not always require the user to specify the data set from which to select the item. If one of the items is in the data set referenced by the selection condition (or one of its ancestors or descendents), or in the last data set selected, that item is chosen. Otherwise, the user is asked to specify the correct data set. Thus, the user is not required to qualify the item every time it is used.

Inquiry indicates that a particular data set must be specified by displaying the following message:

```
#WHICH DATA SET?
1  <data set 1>
2  <data set 2>
   .
   .
   .
n  <data set n>
```

In the preceding list, <data set 1> through <data set n> are the names of the data sets that contain the duplicated item name. The user must then input the number corresponding to the desired data set; this process is known as "qualification." Until a different data set is requested, Inquiry continues to use the data set specified.

If the Inquiry user knows that a particular item is in more than one data set, the item can be explicitly qualified by placing the word OF before the name of the desired data set. For example, if the item EMP-NO were in both the data set EMP and the data set ADR, one of the following qualification clauses would specify which item EMP-NO was desired:

```
EMP-NO OF EMP

EMP-NO OF ADR
```

Item And Value Display

Once a record has been selected, the DISPLAY command may be used to display items within the record. (This command is described in detail under "DISPLAY Command" in Section 3.)

In addition, DISPLAY can cause Inquiry to function like a calculator by displaying the results of arithmetic expressions and certain arithmetic functions involving constants. When Inquiry is used in this way, it is said to be in "calculator mode."

Examples:

A record is selected for an employee in the EMP data set:

```
-> SELECT EMP-FNAME = "JAIME"

   #
```

Next, the item containing the salary for the employee is displayed:

```
-> DISPLAY EMP-SALARY

   EMP-SALARY
   ----------
      2450.00
   #
```

Next, the employee's number, Social Security number, and date of hire are displayed:

```
-> DISPLAY EMP-NO, EMP-SSN, EMP-DATE-HIRED

   EMP-NO     EMP-SSN   EMP-DATE-HIRED
   ------  ---------  --------------
        9  364277866           650714
   #
```

Finally, all items in EMP are displayed for the employee:

```
-> DISPLAY ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     9  JOYCE       JAIME       364277866          650714
   2:TRUE              2450.00
   #
```

The following example illustrates the Inquiry calculator mode. The DISPLAY command in the example lists the value of the EXP function for the given arithmetic expression. (EXP gives the value of "e" raised to the power of the given argument; refer to <arithmetic function> under "Common Syntactic Items" in Section 3.)

```
-> DISPLAY EXP((5/7400) * 12)

                  EXP
   ----------------
        1.00814106784
#
```

Combining Record Selection And Item Display

Record selection and item display can be combined in a single process by appending a selection condition or structure designation to a DISPLAY command. If a selection condition is included, the word AT, WHERE, or IN, or the commercial "at" sign (@), must separate the item name from the selection condition. If a structure is designated, the word USING or VIA must separate the item name from the structure name.

Using this form of item display, all records that meet the selection condition are displayed, so that NEXT need not be specified in order to display each record.

If the specified selection condition requires qualification, the clause specifying the desired data set must appear immediately before the AT, IN, WHERE, or specification.

The user may control the format of displayed output. (Refer to "Output Formatting" in this section.)

Examples:

The following DISPLAY command includes a selection condition specifying that the required information is to be displayed only for employees hired after January 1, 1970:

```
-> DISPLAY EMP-LNAME,EMP-FNAME,EMP-SALARY AT EMP-DATE-HIRED GTR 700101

EMP-LNAME    EMP-FNAME    EMP-SALARY
----------   ----------   ----------
FIELDS       JON             2250.00
HARDWICK     ALICE           3050.00
DEKKER       THOMAS          1950.00
CHAPMAN      GEORGE          2805.00
BURNS        GEORGINA        1750.00
#
```

In the following example, the selection condition designates the subset JOB-MNG-SUBSET, and only records in this subset are displayed:

```
-> DISPLAY JOB-TITLE VIA JOB-MNG-SUBSET

JOB-TITLE
------------------------------
VICE PRESIDENT
CONTROLLER
ADM ASSIST
SALES MANAGER
PROJECT MANAGER
#
```

EMBEDDED STRUCTURES

Data sets may appear as items within other data sets; such data sets are known as "embedded" data sets.

To display an item in an embedded data set, a record from the data set in which the data set is embedded (known as the "master" data set) must first be selected. Records must be selected in ascending hierarchical sequence; that is, for each level of embedding, a record must first be selected at the next higher level of embedding before a record from the lower level may be displayed. A comma (,), the word THEN, or a comma followed by the word THEN must be used to separate the SELECT and DISPLAY commands for each level of nesting.

Because structures must be selected in an ascending hierarchical sequence, a DISPLAY command may reference any structure previously selected at a higher level of nesting.

If items in embedded data sets require qualification, the qualification clauses may be included in DISPLAY command specifications.

An embedded standard or compact data set can only be accessed through a set. In Inquiry, a set used to access an embedded data set must be specified by the word VIA or USING, followed by the name of the set.

Examples:

In the following example, a record is first selected from the master data set EMP, and then all EMP-ED records that are embedded in that EMP record are displayed (Inquiry is instructed to use ED-DATE-SET by the "VIA" specification):

```
-> SELECT AT EMP-LNAME = "MITTY" THEN DISPLAY VIA ED-DATE-SET

   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 540615  UNIV OF RHODE ISLAND                 5227  ENGLISH
   2:  2.00
   #
```

In the following example, selection conditions for the master data set EMP indicate the records of EMP from which the specified embedded ED-DATE-SET records are displayed:

```
-> SELECT EMP-SALARY > 2500 THEN DISPLAY ED-DATE-SET   ED-GPA > 2.5

   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 450615  UNIV OF SO CAL                        2125  HISTORY
   2: 4.00
   1: 650615  UCLA                                  2238  MBA
   2: 4.00
   #
```

USE OF FUNCTIONS

Boolean functions and arithmetic functions may be used in SELECT and DISPLAY commands. (These functions are defined in more detail under Functions in Section 3.)

Boolean Functions

The Boolean functions ANY and ALL allow the specification of conditions involving embedded structures. The function ANY (<selection condition>) yields a value of TRUE if any record embedded in a master data set satisfies the specified selection condition and a value of FALSE otherwise. The function ALL (<selection condition>) yields a value of TRUE if all records embedded in a master data set satisfy the specified selection condition and a value of FALSE otherwise. The specified selection condition may be complex.

Examples:

The following example illustrates the use of the Boolean function ANY. The DISPLAY command displays the names and records in the embedded data set ED-DATE-SET for employees in the EMP data set who have attained a grade-point average of 4.0 at any of the schools they have attended:

```
-> DISPLAY EMP-NAME AT ANY (ED-GPA = 4.0) THEN DISPLAY ED-DATE-SET

  EMP-LNAME    EMP-FNAME
  ----------   ----------
  FIELDS       JON
1:ED-DATE   ED-INST-NAME                    ED-INST-CODE  ED-COURSE
2:ED-GPA
-------------------------------------------------------------------
1: 610101   UNIV OF MARYLAND                      5266   SOC SCI
2: 2.89
1: 650615   KENT STATE                           2123   PSYCHOLOGY
2: 3.77
1: 690612   UCLA                                 2238   BUS ADM
2: 4.00
  EMP-LNAME    EMP-FNAME
  ----------   ----------
  HARDWICK     ALICE
```

```
1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
2:ED-GPA
 ------------------------------------------------------------------
1: 450615  UNIV OF SO CAL                          2125  HISTORY
2: 4.00
1: 650615  UCLA                                    2238  MBA
2: 4.00
#
```

The following example illustrates the use of the Boolean function ALL. The DISPLAY command displays the names and salaries of all employees in the EMP data set whose major subject was Philosophy at all schools attended, then displays the records in the embedded data set ED-DATE-SET for these employees:

```
-> DISPLAY EMP-NAME, EMP-SALARY AT ALL (ED-COURSE="PHILOSOPHY") THEN%
   #%
-> DISPLAY ED-DATE-SET

 EMP-LNAME    EMP-FNAME    EMP-SALARY
 ----------   ----------   ----------
 DEKKER       THOMAS          1950.00
1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
2:ED-GPA
------------------------------------------------------------------
1: 630615  UNIV OF MONTANA                         2943  PHILOSOPHY
2: 3.00
1: 670623  UNIV OF IDAHO                          1637  PHILOSOPHY
2: 3.00
#
```

Arithmetic Functions

Arithmetic functions specify arithmetic operations involving structures, which produce values that may be used in SELECT or DISPLAY command specifications. Arithmetic functions are of two kinds: data set functions and current record functions.

The following functions are the data set functions:

COUNT	MAX (maximum)	MSQ (mean square)
SUM	MIN (minimum)	VAR (variance)
AVG (average)	SSQ (sum of squares)	STD (standard deviation)

The following functions are the current record functions:

SCALE	TAN (tangent)	EXP (exponential)
ABS (absolute value)	ARCSIN (arcsine)	LN (natural log)
SIN (sine)	ARCCOS (arccosine)	LOG (logarithm base 10)
COS (cosine)	ARCTAN (arctangent)	
SQRT (square root)		

To compute the values for data set arithmetic functions, Inquiry may need to access every record of the specified data set; thus, a large amount of time may be used in producing results. If the amount of time required to produce the results exceeds the user's expectations, the user can discontinue processing by by entering the CANDE ?AX command. The system then displays the following message and awaits additional input:

```
#STOPPED
```

Examples:

The following example displays the names of all employees in the data set EMP whose monthly salaries are greater than or equal to the average of all employees' salaries:

```
-> DISPLAY EMP-NAME AT EMP-SALARY GEQ AVG (EMP-SALARY)

  EMP-LNAME   EMP-FNAME
  ----------  ----------
  FIELDS      JON
  HARDWICK    ALICE
  CHAPMAN     GEORGE
  JOYCE       JAIME
#
```

The following example displays the last name of each employee in EMP and the difference between the employee's salary and the average salary of the employees in EMP:

```
-> DISPLAY EMP-LNAME, ABS(EMP-SALARY - AVG(EMP-SALARY)) VIA EMP

  EMP-LNAME                    ABS
  ----------  ------------------
  FIELDS            2.60444444418
  HARDWICK           802.60444444
  JONSON            475.835555556
  DEKKER            297.395555556
  CHAPMAN            557.60444444
  MARLOWE           47.3955555558
  BURNS             497.395555556
  MITTY             247.395555556
  JOYCE             202.604444444
#
```

The following example displays the same information as the preceding example, except that the difference between each employee's salary and the average is rounded to three decimal places:

```
-> DISPLAY EMP-LNAME, SCALE(ABS(EMP-SALARY - AVG(EMP-SALARY)),3)%
   #%
-> VIA EMP

  EMP-LNAME                   SCALE
  ----------  ------------------
  FIELDS                      2.604
  HARDWICK                  802.604
  JONSON                    475.836
  DEKKER                    297.396
  CHAPMAN                   557.604
  MARLOWE                    47.396
  BURNS                     497.396
  MITTY                     247.396
  JOYCE                     202.604
#
```

VIRTUAL ITEMS

Inquiry may be used to compute or display a value that is not physically present in the data base. Such an item, known as a "virtual" item, is created using the VIRTUAL command. (Refer to "Virtual Command" in Section 3.)

Any form of arithmetic expression is allowed to the right of the equal sign (=) in the VIRTUAL command. The rule is that the items specified in a VIRTUAL command must yield a result when the virtual item is referenced.

Virtuals cannot be recursive, either directly or indirectly. If an attempt is made to evaluate a virtual that is recursive, an error message results.

System efficiency is improved when arithmetic functions are associated with virtual identifiers. If an arithmetic function appears in a display list or selection condition, the system must evaluate the function each time the function is referenced. By contrast, if an arithmetic function is associated with a virtual identifier, the system "remembers" whether the function has been evaluated and, if so, also remembers the value of the function.

The system recognizes when the value of an arithmetic function associated with a virtual identifier becomes meaningless and re-evaluates this value, if necessary.

A virtual cannot be cleared until all virtuals (if any) that reference that virtual have been cleared. The only exception to this rule is during use of the CLEAR VIRTUALS command which clears all virtuals. Thus, if VIRTUAL A = 10 and VIRTUAL B = A + 5, virtual A cannot be cleared until virtual B is cleared or the CLEAR VIRTUALS command is used.

Examples:

In the following example, a virtual item, SKILLINDEX, is created based on the typing speed and shorthand speed of the employees in the data base:

```
-> VIRTUAL SKILLINDEX = (TYPE-SPEED + SHORTHAND-SPEED)/100
```

The virtual item SKILLINDEX is then displayed for the specified job type:

```
-> DISPLAY JOB-TITLE, SKILLINDEX AT JOB-TYPE = 1

JOB-TITLE                          SKILLINDEX
-------------------------------  ------------------
RECEPTIONIST                                   1
SECRETARY                                    1.7
EXEC SECRETARY                               0.6
#
```

In the following example, a virtual item, OUR-GPA, is created based on the grade-point average of employees in the data base:

```
-> VIRTUAL OUR-GPA = ED-GPA * 2.5

#
```

The virtual item OUR-GPA is then displayed for the specified employee in EMP along with the university that the employee attended and the employee's major field of study:

```
-> SELECT EMP AT EMP-LNAME = "MARLOWE" THEN DISPLAY ED-INST-NAME,%
#%
-> ED-COURSE, OUR-GPA VIA ED-DATE-SET

ED-INST-NAME                    ED-COURSE              OUR-GPA
------------------------------  ----------  ------------------
UCLA                            ACCT                      7.5
#
```

UNDEFINED ITEMS

References can be made in selection conditions, display lists, or arithmetic expressions that cause the values of items to be undefined. Item values are undefined in the following situations:

1. The value of an item is NULL (as defined in DASDL).

2. The value of a function is undefined.

3. Variable-format records are used, and a record is loaded in which the specified item does not exist.

4. Evaluation of an arithmetic expression results in division by zero.

5. An attempt is made to generate a number that exceeds the hardware capacity (integer overflow, exponent overflow, or exponent underflow).

6. A subscript is expressed as an arithmetic expression that yields an undefined value on evaluation. In this case, the system displays the following error message:

   #SUBSCRIPT UNDEFINED

If an item with an undefined value is displayed, two hyphens are displayed instead of the value of the item.

Example:

In the following example, the item TYPE-SPEED is undefined for the specified JOB-TYPE:

```
-> DISPLAY JOB-TITLE, JOB-TYPE, TYPE-SPEED  AT JOB-TYPE = 2

  JOB-TITLE                      JOB-TYPE  TYPE-SPEED
  ------------------------------ --------  ----------
  SALESMAN                              2          --
  SALES MANAGER                         2          --
  PROJECT MANAGER                       2          --
#
```

RELATED RECORDS ACROSS DATA SETS

Ordinarily, Inquiry selects records or displays items from a single data set at a time. However, if two unrelated data sets both contain a common item, Inquiry is able to access information in both data sets if the common items for the data sets are set equal to one another.

The SET command is used to associate a specified Inquiry command with a particular structure. The specified command is not executed immediately, but Inquiry is able to remember it. The REPEAT command is used to execute a command that has been remembered in this way. (Refer to SET Command and REPEAT Command in Section 3 for descriptions of this process.)

Examples:

In the following example, the item EMP-NO in the EMP data set and the corresponding item ADR-EMP-NO in the ADR data set are set equal to each other. In this way, the name, Social Security number, address, and zip code of each employee with an address in the data base can be displayed:

```
-> SET ADR TO DISPLAY EMP-NAME, EMP-SSN, ADR-STREET-NO, ADR-ZIP %

   #%

-> AT ADR-EMP-NO = EMP-NO

   #

-> SELECT EMP, THEN REPEAT ADR

   1:EMP-LNAME    EMP-FNAME      EMP-SSN   ADR-STREET-NO
   2:ADR-ZIP
   ------------------------------------------------------------------
   1:FIELDS       JON          533733557   5 MAIN ST NEWTOWN PA
   2:  56662
   #

-> NEXT EMP

   1:EMP-LNAME    EMP-FNAME      EMP-SSN   ADR-STREET-NO
   2:ADR-ZIP
   ------------------------------------------------------------------
   1:HARDWICK     ALICE        647938292   45 ALEXANDER RD DU BOIS PA
   2:  50007
   #
```

In the following example, a REPEAT command is included among the commands associated with EMP to establish "automatic looping" between EMP and ADR. That is, every time the commands associated with EMP are executed, the commands associated with ADR are executed, as well:

```
-> SET ADR TO DISPLAY ADR-STREET-NO, ADR-ZIP AT ADR-EMP-NO = EMP-NO

#

-> DISPLAY EMP-NAME, EMP-NO VIA EMP AT EMP-NO < 4 THEN REPEAT ADR

  EMP-LNAME   EMP-FNAME   EMP-NO
  ----------  ----------  ------
  FIELDS      JON              1

  ADR-STREET-NO                    ADR-ZIP
  -------------------------------  -------
  5 MAIN ST NEWTOWN PA               56662

  EMP-LNAME   EMP-FNAME   EMP-NO
  ----------  ----------  ------
  HARDWICK    ALICE            2

  ADR-STREET-NO                    ADR-ZIP
  -------------------------------  -------
  45 ALEXANDER RD DU BOIS PA         50007

  EMP-LNAME   EMP-FNAME   EMP-NO
  ----------  ----------  ------
  JONSON      BEN              3

  ADR-STREET-NO                    ADR-ZIP
  -------------------------------  -------
  678 ARCADIA BLVD ROSSITER PA       57667
#
```

DATA BASE MODIFICATION

The Inquiry UPDATE, CREATE, and DELETE commands may be used to modify the contents of a data base.

Update

The UPDATE command changes the values of items in the currently selected record. Several items, together with their new values, may be specified in a single UPDATE command; however, all specified items must belong to the same record.

When changing the number of occurrences of an occurring item that has OCCURS DEPENDING ON specified in DASDL, the user must ensure that the "depending on" item is properly updated. Failure to maintain the value for the "depending on" item can result in the loss of new values for occurring items.

Because Inquiry remembers the list of updated items for each data set, the user need not re-enter the same UPDATE command if the same change is to be made to several records in the data set. Instead, the UPDATE command may specify the data set name; Inquiry then performs the change indicated by the most recently entered UPDATE command for that data set.

If the user forgets the contents of the current update list or wishes to edit the current update list, the RECALL command may be executed to display the current update list for any structure. (Refer to "RECALL Command" in Section 3.)

Inquiry may be generated without the UPDATE feature in order to prevent unauthorized data base modification. (Refer to "Inquiry Generation" in Appendix C.)

Examples:

In the following example, the record for the first employee in the data set EMP with security clearance is selected and displayed:

```
-> SELECT EMP AT EMP-SEC-CLEAR

   #

-> DISPLAY ALL
```

```
1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:     1  FIELDS       JON         533733557          720312
2:TRUE              2250.00
#
```

The UPDATE command is then used to change the employee's security clearance to FALSE and the employee's salary to 2000:

```
-> UPDATE EMP-SEC-CLEAR = FALSE, EMP-SALARY = 2000

   SELECTED RECORD OF EMP UPDATED
   #


-> DISPLAY ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   -------------------------------------------------------------------
   1:     1  FIELDS       JON         533733557          720312
   2:FALSE             2000.00
   #
```

The record for the next employee in EMP with security clearance is then selected and displayed:

```
-> NEXT

   #


-> DISPLAY ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   -------------------------------------------------------------------
   1:     2  HARDWICK    ALICE       647938292          701214
   2:TRUE              3050.00
   #
```

The UPDATE command is then used to change the employee's last name from "HARDWICK" to "KRANTZ":

```
-> UPDATE EMP-LNAME = "KRANTZ"
```

```
   SELECTED RECORD OF EMP UPDATED
   #
-> DISPLAY ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     2  KRANTZ       ALICE       647938292          701214
   2:TRUE               3050.00
   #
```

Create And Delete

A new record is added to a data base using the CREATE command (refer to "CREATE Command" in Section 3). If no data set name is specified in the CREATE command, Inquiry uses the items specified in the CREATE command to decide which data set is to contain the newly created record. The newly created record remains the currently selected record.

The CREATE command can add a record to an embedded structure; however, such a record can be created only if a record is currently selected for the master structure.

Any items omitted from a CREATE command are assigned the initial values specified by DASDL. If any required items are left null, the newly created record cannot be stored, and an error message is displayed.

Records may be deleted from a data base using the DELETE command (refer to "DELETE Command" in Section 3). A DELETE command must specify the data set from which the record is to be deleted.

The normal DMSII rules for performing record deletion also apply to Inquiry record deletion. That is, before a record that contains embedded data sets can be deleted, all of its embedded records must first be deleted.

Inquiry may be generated without the CREATE and DELETE commands in order to prevent unauthorized data base modification. (Refer to "Inquiry Generation" in Appendix C.)

Examples:

In the following example, a new record is created for the data set ADR and then displayed:

```
-> CREATE ADR WITH ADR-EMP-NO=11, ADR-STREET-NO = "14 HILL MISSOULA%
   #%
->  MT", ADR-ZIP = 87143

   RECORD OF ADR CREATED
   #

-> DISPLAY ALL

   ADR-EMP-NO  ADR-STREET-NO                    ADR-ZIP
   ----------  -------------------------------  -------
           11  14 HILL MISSOULA MT                87143
   #
```

The newly created record is then deleted:

```
-> DELETE ADR

   SELECTED RECORD OF ADR DELETED
   #

-> SELECT ADR AT ADR-EMP-NO = 11

   #NONE
   #
```

In the following example, a record of the data set ADR is selected and then deleted:

```
-> SELECT ADR AT ADR-STREET-NO = "5 MAIN ST NEWTOWN PA"

   #

-> DELETE ADR

   SELECTED RECORD OF ADR DELETED
   #

-> SELECT ADR AT ADR-STREET-NO = "5 MAIN ST NEWTOWN PA"

   #NONE
```

The following example illustrates the deletion of a record from an embedded data set. First, a record of the data set EMP is selected, and the corresponding record in the embedded set ED-DATE-SET is displayed:

```
-> SELECT EMP THEN DISPLAY ED-DATE-SET

   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 610101  UNIV OF MARYLAND                        5266  SOC SCI
   2:  2.89
   1: 650615  KENT STATE                              2123  PSYCHOLOGY
   2:  3.77
   1: 690612  UCLA                                   2238  BUS ADM
   2:  4.00
   #
```

Next, a record in the embedded data set EMP-ED is selected using the embedded set ED-DATE-SET:

```
-> SELECT ED-DATE-SET AT ED-INST-NAME = "UCLA"

   #
```

The selected record in EMP-ED is then deleted:

```
-> DELETE EMP-ED

   SELECTED RECORD OF EMP-ED DELETED
   #

-> SELECT ED-DATE-SET AT ED-INST-NAME = "UCLA"

   #NONE
   #
```

Modification Of Audited Data Bases

When Inquiry create, delete, and update operations are performed on an audited data base, they execute as a single transaction with a sync-point forced on the ENDTRANSACTION. (Refer to the User Language Interface SOG for a discussion of transactions.) This method of access has two advantages:

1. Other programs that may be using the data base simultaneously with Inquiry are not suspended while Inquiry waits for a response from a terminal.

2. Inquiry transactions are not backed out when a Halt/Load occurs or a simultaneous user program aborts.

If an error occurs on an Inquiry transaction, the error is reported and the data base is not modified.

OUTPUT CONTROL AND FORMATTING

The following paragraphs illustrate ways in which Inquiry is used to control the format of displayed output.

DISPLAY Command

If a selection expression is included in a DISPLAY command, the DISPLAY command can display information for as many records satisfying the selection condition as can fit on the output device. This form of the DISPLAY command allows the user to control the amount of output displayed at one time by including a limit in the DISPLAY command. (Refer to Option 2 under DISPLAY Command in Section 3.) If the user then enters NEXT or a blank, Inquiry responds by selecting the next record and displaying records according to the previously entered specifications. This procedure may be repeated until all records that satisfy the selection condition have been displayed.

If output is displayed on a terminal, Inquiry automatically sets the limit for displayed output to the screen size of the terminal.

If a new Inquiry command is entered while a display operation is in progress, the remaining displayed output is discarded and the system processes the new command.

Examples:

The following DISPLAY command specifies that records meeting the selection condition are to be displayed two at a time:

```
-> DISPLAY 2 ADR-INFO AT ADR-EMP-NO < 7

   ADR-STREET-NO                    ADR-ZIP
   -------------------------------  -------
   45 ALEXANDER RD DU BOIS PA         50007
   678 ARCADIA BLVD ROSSITER PA       57667
   #
-> NEXT

   ADR-STREET-NO                    ADR-ZIP
   -------------------------------  -------
   2345 HUNTINGTON ST READING PA      50007
   6 OLIVE AVE BELVEDERE NJ           56432
```

```
-> NEXT

   ADR-STREET-NO                    ADR-ZIP
   ------------------------------   -------
   23 PHILIPS ST APT 2 MARLENA GA     42323
   #NO MORE
   #
```

Embedded Structures

In a DISPLAY command involving embedded structures, the amount of uninterrupted output can be controlled on a structure basis. Output limit numbers also can be used to selectively bypass some of the output.

Output control can be imposed at each level of nesting.

In certain instances, stopping on a limit for one structure coincides with stopping on a limit for another structure. In these cases, the user may have to enter NEXT twice to resume output.

Examples:

In the following example, the first record in the data set EMP is selected. The colleges (contained in the embedded data set EMP-ED) attended by that employee where the employee attained a grade-point average greater than 3.5 are then displayed, one at a time:

```
-> SELECT EMP THEN DISPLAY 1 ED-INST-NAME WHERE ED-GPA > 3.5

   ED-INST-NAME
   ----------------------------
   CARLETON
   #
-> NEXT

   ED-INST-NAME
   ----------------------------
   UCLA
   #

-> NEXT

   #
```

The second of the preceding NEXT commands was requesting the next ED-INST-NAME for this same employee. However, at this point no more embedded items satisfy the selection condition for this EMP. Because the implicit limit for the SELECT command is 1, the NEXT command must be entered again to select the next record from EMP and display the desired information for that record:

```
-> NEXT

   ED-INST-NAME
   ----------------------------
   UNIV OF SO CAL
   #
```

In the following example, a record is selected from EMP, and a college where the employee's major was not Philosophy is displayed:

```
-> SELECT EMP THEN DISPLAY 1 ED-INST-NAME WHERE ED-COURSE NEQ%
   #%
->  "PHILOSOPHY"

   ED-INST-NAME
   ----------------------------
   UNIV OF MARYLAND
   #
```

Then, the next record from EMP is selected and the same information is displayed for the employee corresponding to the newly selected record:

```
-> NEXT EMP

   ED-INST-NAME
   ----------------------------
   UNIV OF SO CAL
   #
```

If NEXT alone had been entered in the preceding example, the next ED-INST-NAME record would have been displayed for the same employee.

Finally, the schools attended by the newly selected employee where the employee's major was not Social Science are displayed two at a time:

```
-> DISPLAY 2 ED-INST-NAME WHERE ED-COURSE NEQ "SOC SCI"
```

```
   ED-INST-NAME
   ----------------------------
   UNIV OF SO CAL
   UCLA
   #

-> NEXT

   #NO MORE
   #
```

Formatting Attributes

The format of the data displayed on the terminal or printer is controlled by the format attributes HEADING, TAB, and SINGLE. The following discussion pertains to output formatting on a terminal; the same formats are displayed with output to a printer. (Refer to TERMINAL Command and PRINTER Command in Section 3.)

Heading

The HEADING attribute causes output to be displayed in the following format:

```
        N1        N2        N3
        --        --        --

        V11       V12       V13

        V21       V22       V23
```

In the preceding format diagram, N1, N2, and N3 are the item names; V11, V12, and V13 are the values associated with those items in record 1; and V21, V22, and V23 are the values of those items in record 2.

HEADING is set by entering the following form of the TERMINAL command:

```
TERMINAL FORMAT HEADING
```

If desired, the following abbreviation may be used for this command:

```
T F H
```

If the abbreviated form is entered, each letter (T F H) must be separated by a space.

In some cases the display list is so large that all item names, or all item values, or both cannot fit on a single line. In these cases, the displayed output has the following format:

```
1:N1      N2        N3    <---!
                              !-- Item names
2:N4      N5        N6    <---!

1:V11     V12       V13   <---!
                              !-- Values, first record
2:V14     V15       V16   <---!

1:V21     V22       V23   <---!
                              !-- Values, second record
2:V24     V25       V26   <---!
```

Association between each item name and the corresponding item value is made by line number and position within the line. For example, item N1 has a value of V11 in the first record and a value of V21 in the second record; item N5 has a value of V15 in the first record and a value of V25 in the second record.

If output is displayed at a terminal and the number of title lines and value lines for a single record exceeds the screen limit, Inquiry sets the TAB formatting attribute and displays output in TAB format from that point onward. Otherwise, all displayed output appears in HEADING format by default.

Examples:

In the following example, items from the data set JOB are displayed, three at a time, in HEADING format:

```
-> T F H

   #

-> D 3 JOB
```

```
1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
3:DRIVING-REQD  JOB-COMMENT
-------------------------------------------------------------------
1:        1         0      9999  VICE PRESIDENT
2:        --                 --  --           --
3:--            --
1:        1         0      4000  ACCOUNTANT
2:        --                 --  --           --
3:--            --
1:        1         0      8888  CONTROLLER
2:        --                 --  --           --
3:--            --
#
```

Tab

The TAB attribute causes displayed output to appear in the following format:

N1 = V11 N2 = V12 N3 = V13

N4 = V14 N5 = V15

In the format diagram above, N1 through N5 represent the item names, and V11 through V15 represent the values associated with the item names.

TAB is set by entering the following form of the TERMINAL command:

```
TERMINAL FORMAT TAB
```

If desired, the following abbreviation may be used for this command:

```
T F T
```

If the abbreviated form is entered, each letter must be separated by a space.

Each "name=value" combination begins at a new tab position, where the tab stops are at every fifth position starting with character position 3.

```
1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
3:DRIVING-REQD  JOB-COMMENT
-------------------------------------------------------------------
1:        1         0      9999  VICE PRESIDENT
2:        --                 --  --           --
3:--            --
1:        1         0      4000  ACCOUNTANT
2:        --                 --  --           --
3:--            --
1:        1         0      8888  CONTROLLER
2:        --                 --  --           --
3:--            --
#
```

Tab

The TAB attribute causes displayed output to appear in the following format:

N1 = V11 N2 = V12 N3 = V13

N4 = V14 N5 = V15

In the format diagram above, N1 through N5 represent the item names, and V11 through V15 represent the values associated with the item names.

TAB is set by entering the following form of the TERMINAL command:

```
TERMINAL FORMAT TAB
```

If desired, the following abbreviation may be used for this command:

```
T F T
```

If the abbreviated form is entered, each letter must be separated by a space.

Each "name=value" combination begins at a new tab position, where the tab stops are at every fifth position starting with character position 3.

N2 = V22

N3 = V23

In this formatting diagram, N1 through N3 represent the item names, and V11 through V23 represent the associated values. SINGLE causes one item name and its associated value to be displayed on each line starting at character position 3. If more than one record is displayed, a blank line separates the displayed records.

SINGLE is set by entering the following form of the TERMINAL command:

TERMINAL FORMAT SINGLE

If desired, the following abbreviation may be used for this command:

T F S

If the abbreviated form is entered, each letter must be separated by a space.

Examples:

In the following example, items from the JOB data set are displayed, three at a time, in SINGLE format:

```
-> T F S

   #


-> D 3 JOB

      JOB-COUNT = 1
      JOB-TYPE = 0
      JOB-CODE = 9999
      JOB-TITLE = VICE PRESIDENT
      TYPE-SPEED = --
      SHORTHAND-SPEED = --
      CRTUSE-REQD = --
      SPANISH-REQD = --
      DRIVING-REQD = --
      JOB-COMMENT = --
```

```
JOB-COUNT = 1
JOB-TYPE = 0
JOB-CODE = 4000
JOB-TITLE = ACCOUNTANT
TYPE-SPEED = --
SHORTHAND-SPEED = --
CRTUSE-REQD = --
SPANISH-REQD = --
DRIVING-REQD = --
JOB-COMMENT = --

JOB-COUNT = 1
JOB-TYPE = 0
JOB-CODE = 8888
JOB-TITLE = CONTROLLER
TYPE-SPEED = --
SHORTHAND-SPEED = --
CRTUSE-REQD = --
SPANISH-REQD = --
DRIVING-REQD = --
JOB-COMMENT = --

#
```

INQUIRY REPORT

The Inquiry REPORT feature produces simple reports from information contained in DMSII data bases.

A number of Inquiry commands are used to specify a report. (Refer to Section 3 for detailed discussions of these commands.)

The REPORT command specifies items and values to be printed on the report, names and positions of columns on the report, and criteria for control breaks and page control.

Selection conditions for records input to the report process may be specified using either the SELECT command or the SET command. Normally, the SET command should be used in order to avoid redundant calls on DMSII functions.

The optional SUMMARY command can be used to cause the report process to gather statistical information, such as column averages, totals, and counts, for items in the report list. The SUMMARY command, if used, must follow the REPORT command.

The optional TITLE command can be used to specify an initial report title page, a heading for the top of each report page, or both. The length of a report-title line cannot exceed the line width of the output device. The column position number indicated in the report specifications added to the column heading size must not exceed the device line size. For reports directed to line printers, the line width is equal to the printer width. For reports directed to hardcopy terminals, the line width is equal to the terminal width. For reports directed to screen terminals, the line width is equal to the terminal width minus one column. In addition, the last 10 columns of the first line of a report title are reserved for the page number. Thus, the length of the first line of a title cannot exceed the column width minus 21.

If more than one report is to be generated during a session, a CLEAR REPORT command should be entered before entering any command for the next report.

The GENERATE command initiates the actual report-generation process. The report can be sent to either the user's terminal or the site printer by specifying TERMINAL or PRINTER, respectively, in the OPTIONS command.

If the report is intended for the site printer, OPTIONS PRINTER should be entered before the GENERATE command is entered, as the formatting of the report is based on the attributes specified in the PRINTER and TERMINAL commands.

When the report is output to a hardcopy terminal and a new page is required, then sufficient blank lines are written to ensure a full page. (The report-generation procedure automatically writes four blank lines at the end of an output page. These lines should be taken into account if the terminal PAGE attribute is set; refer to TERMINAL Command in Section 3.) Then, if the terminal SCREEN attribute is TRUE, the system stops until a blank character is input. If SCREEN is FALSE, then four more blank lines are printed before the next page is started.

The RECALL command can be used to display any previously entered REPORT, SUMMARY, or TITLE command. The recalled command can then be modified using the EDIT and REPEAT commands. If a REPORT command is modified in this way, then any SUMMARY command associated with the report that contains items that were added or deleted from the original REPORT command should be modified, as well. Inquiry displays the following message if REPORT command specifications have been modified without modifying an associated SUMMARY command:

```
**CAUTION-CHECK CONTROL AND/OR REPORT SUMMARY
```

The various statements used to generate a report should be entered using a multiple-command DEFINE. In this way, report definitions can be saved between Inquiry sessions.

Example:

In the following example, a report is generated listing employees in the data set EMP that were hired before 1962. First, a selection condition for these employees is entered:

```
-> SELECT EMP AT EMP-DATE-HIRED < 620101
```

Next, the REPORT command is used to specify the items (each employee's last name, hiring date, employee number, and salary) to appear in the report:

```
-> REPORT EMP-LNAME, EMP-DATE-HIRED, EMP-NO, EMP-SALARY
   #
```

A TITLE command is entered to specify a heading for each page of the report:

```
-> TITLE PAGE "20 YEAR EMPLOYEES"

   #
```

Finally, the report is generated using the GENERATE command:

```
-> GENERATE REPORT VIA EMP

   #STARTING REPORT

                              20 YEAR EMPLOYEES                    PAGE 1
    EMP-LNAME  EMP-DATE-HIRED EMP-NO EMP-SALARY
    ---------- -------------- ------ ----------
    MARLOWE            551225      6    2200.00
    MITTY              540303      8    2000.00

    # REPORT COMPLETED
    #
```

In the following example, the original report specifications are recalled and edited. A REPEAT command is entered to replace the original specifications with the edited specifications, and a new report is generated:

```
-> RECALL REPORT

   REPORT EMP-LNAME,EMP-DATE-HIRED,EMP-NO,EMP-SALARY
   #

-> EDIT /EMP-DATE-HIRED/EMP-FNAME

   REPORT EMP-LNAME,EMP-FNAME,EMP-NO,EMP-SALARY
   #

-> REPEAT

   #


-> GENERATE REPORT VIA EMP

   # STARTING REPORT
```

```
                          20 YEAR EMPLOYEES                    PAGE 1
 EMP-LNAME   EMP-FNAME   EMP-NO EMP-SALARY
 ----------- ---------- ------ ----------
 MARLOWE     PHILLIP         6    2200.00
 MITTY       WALTER          8    2000.00

# REPORT COMPLETED
#
```

In the following example, the SUMMARY command is used to specify that the number of employees in the report is printed at the end of the report:

```
-> SUMMARY REPORT COUNT(EMP-LNAME)

   #

-> GENERATE REPORT VIA EMP

   # STARTING REPORT
                          20 YEAR EMPLOYEES                    PAGE 1
 EMP-LNAME   EMP-FNAME   EMP-NO EMP-SALARY
 ----------- ---------- ------ ----------
 MARLOWE     PHILLIP         6    2200.00
 MITTY       WALTER          8    2000.00
                       .
                       .
                       .
                          20 YEAR EMPLOYEES                    PAGE 2
   SUMMARY FOR REPORT

   EMP-LNAME :COUNT=2

   # REPORT COMPLETED
   #
```

In the following example, the REPORT command specifies the beginning column and column length for the items specified as column headings in a new report. Inquiry displays a warning message because no modification has been made to the associated SUMMARY command:

```
-> REPORT EMP-LNAME AT 2 FOR 20, EMP-FNAME AT 30 FOR 10, EMP-NO AT 45%
   %#
->  FOR 10, EMP-SALARY AT 60 FOR 10

   **CAUTION-CHECK CONTROL AND/OR REPORT SUMMARY
   #

-> GENERATE REPORT VIA EMP
```

```
# STARTING REPORT

                          20 YEAR EMPLOYEES                    PAGE 1
EMP-LNAME                    EMP-FNAME          EMP-NO       EMP-SALARY
-------------------          ----------     ----------       ----------
MARLOWE                      PHILLIP                 6          2200.00
MITTY                        WALTER                  8          2000.00
                        .
                        .
                        .
                           20 YEAR EMPLOYEES                   PAGE 2
SUMMARY FOR REPORT

EMP-LNAME :COUNT=2

# REPORT COMPLETED
#
```

INQUIRY EXTRACT

The Inquiry EXTRACT feature allows items in selected records to be extracted to a standard data file.

A number of commands are used to specify an extract file. (Refer to Section 3 for detailed descriptions of these commands.)

The EXTRACT command specifies the items to be extracted, allows changes to be made to the size and format of items, and positions items in the extracted record. A record layout is produced indicating the item name, item type and size, and item offset (in digits) in the record.

As in the Inquiry REPORT feature, the SET or SELECT command specifies selection conditions for records to appear in the extract file, and the extract process is initiated using a GENERATE command. A single GENERATE command may be used to generate both an extract file and a report. Because items for the extract file and items for the report are specified in different commands, different items may be extracted and reported using the same GENERATE command.

The optional FILE command may be used to set commonly used file attributes for the extract file. If the INTNAME attribute is set using this command, Work Flow Language (WFL) file-equation attributes may be attached to the extract file.

Examples:

The following EXTRACT command creates an extract file based on items from the data set EMP according to the following specifications:

1. The items EMP-LNAME, EMP-SSN, and EMP-SALARY, plus the result of the arithmetic expression EMP-SALARY * 12, are extracted. EMP-LNAME is, by default, extracted in ALPHA format. The arithmetic expression result is, by default, extracted in REAL format. EMP-SSN and EMP-SALARY are, according to the specification in the EXTRACT command, extracted in NUMBER format.

2. EMP-SSN is extracted with a precision of 9. EMP-SALARY is extracted with a precision of 5 and a scale factor of 4.

3. In the event of a null error for EMP-SALARY, the value for EMP-SALARY in the extract file is all bits ON. In the event of

a fault error for the arithmetic expression EMP-SALARY * 12, the resulting value is all bits ON, and a warning message is generated.

```
-> EXTRACT EMP-LNAME, EMP-SSN N (9), EMP-SALARY N (5,4) [NULL=HIGH],%
   #%
-> EMP-SALARY * 12 [FAULT=HIGH WARN]

   # EXTRACT RECORD SIZE WILL BE 4 WORDS UNLESS CHANGED BY FILE COMMAND
   # EXTRACT FILE LAYOUT (EXTRACT/DIV68 ON DISK.)
     ITEM              TYPE          OFFSET
   EMP-LNAME         ALPHA(10)          0
   EMP-SSN           NUMBER(9)         20
   EMP-SALARY        NUMBER(5,4)       29
   AEXP1             REAL              34
   #
```

The following GENERATE command is then executed to create the extract file:

```
-> GENERATE EXTRACT VIA EMP

   # STARTING EXTRACT
   # EXTRACT COMPLETED
   #
```

3. INQUIRY COMMANDS

INTRODUCTION

This section describes the syntax and semantics of each Inquiry input command. All Inquiry commands except SELECT and DISPLAY are discussed alphabetically in this section. Because of their close relationship and their importance to the Inquiry program, the SELECT and DISPLAY commands are discussed together at the beginning of the section, immediately following the description of common syntactic items.

SUMMARY OF INQUIRY COMMANDS

A brief description of each Inquiry command is given in the following list.

ATTACH

Combines statements for an embedded data set with statements for its master data set to establish automatic looping between the two structures.

CLEAR

Discards DEFINE items, VIRTUAL items, and/or GENERATEd subsets.

CREATE

Creates a new record to be added to a data set.

DEFINE

Assigns a name to a combination of one or more Inquiry constructs (for example, identifiers, display lists, selection conditions, or commands).

DELETE

Removes the currently selected record from a data set.

DETACH

Separates an embedded data set from its master data set to prevent automatic looping between the two structures.

DISPLAY

Displays items of a selected record or the value of an arithmetic expression involving numeric constants.

EDIT

Modifies contents of Inquiry commands in the current buffer.

EXTRACT

Specifies items to be included in an extract file and controls the format of these items in the report.

FILE

Displays or sets commonly used attributes for extract files.

GENERATE

Creates a report, an extract file, or a temporary subset of a data set.

HELP

Displays syntax or semantics for Inquiry commands or diagnostic information related to the last error encountered.

NEXT

Causes Inquiry to resume record selection and item display from the most recent stopping point.

OPTIONS

Controls destination and contents of output from DISPLAY commands.

PRINTER

Displays or alters printer-file attributes assigned by Inquiry.

QUIT

Terminates the Inquiry session.

RECALL

Loads the current buffer with a previous Inquiry command and displays this command on the terminal.

REDEFINE

Changes the text associated with a define or virtual.

REPEAT

Executes the command in the current buffer.

REPORT

Specifies items to be listed in a report and controls the format of the report.

RESTORE

Reloads previously saved declarations.

SAVE

Stores the text of define items, virtual items, and generated subsets in a file on disk for later use.

SELECT

Locates records that satisfy selection conditions.

SET

Establishes the text of a DISPLAY, REPEAT, SELECT, CREATE, SORT, or UPDATE command, a SORT option, or a limit option for a given data set. Can also be used to modify or delete the text of the most recently entered such construct for a given data set.

SHOW

Displays specified parts of a data base description, contents of the current buffer, define items, virtual items, or generated subset specifications.

SORT

Displays or modifies core and disk parameters used by the SORT option.

SUMMARY

Provides statistical information about items contained in a report list.

TERMINAL

Displays or alters terminal file attributes.

TITLE

Defines the title characteristics of a report.

UPDATE

Modifies items in a selected record and stores updated record in the data set.

VIRTUAL

Creates virtual items (items that are functions of other items).

COMMON SYNTACTIC ITEMS

The following paragraphs give descriptions and syntax for a number of constructs that commonly appear in Inquiry commands.

<selection condition>

A <selection condition> specifies conditions that a record must satisfy if that record is to be selected.

A simple <selection condition> is a generalized Boolean expression consisting of Boolean items and primaries. Boolean items in a selection condition must appear in the record to be selected. Boolean primaries yield a truth value of either TRUE or FALSE on evaluation; all items referenced in Boolean primaries must be contained in the current record or in records previously selected at a higher nesting level.

Parentheses and the logical operators AND, OR, and NOT can be used in conjunction with simple <selection condition>s to form complex <selection condition>s. The following order of precedence controls the sequence in which <selection condition>s are evaluated.

1. Arithmetic Expressions.

2. Relational Operators.

3. NOT.

4. AND.

5. OR.

When operators have the same precedence, the operations are performed in order from left to right. Parentheses are used to override the usual order of evaluation.

<selection condition>s may also contain relationships involving alpha items, alpha literals, and relational operators. Care must be taken that alpha literals beginning with letters are not mistaken for

identifiers; by default, Inquiry interprets such alpha literals as identifiers if any confusion arises.

All items appearing in <selection condition>s must be contained in currently selected records, in the record being selected, or (if the Boolean functions ANY or ALL are used in the <selection condition>) in a record embedded in the record being selected.

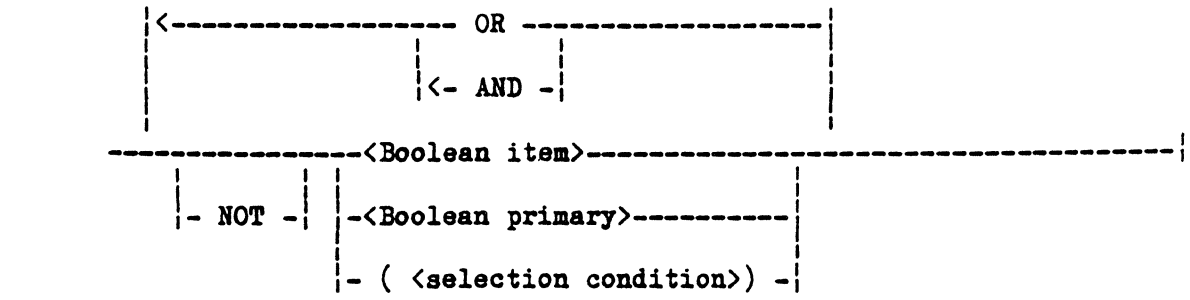
Items in a <selection condition> may be tested to determine if they have NULL values. Only items that may take NULL values may be tested. If an attempt is made to test an item that cannot take a NULL value, the following error message is displayed:

```
ERR: NULL
#TEST FOR NULL IS SUPERFLUOUS--ITEM CAN NOT BE NULL.
#
```

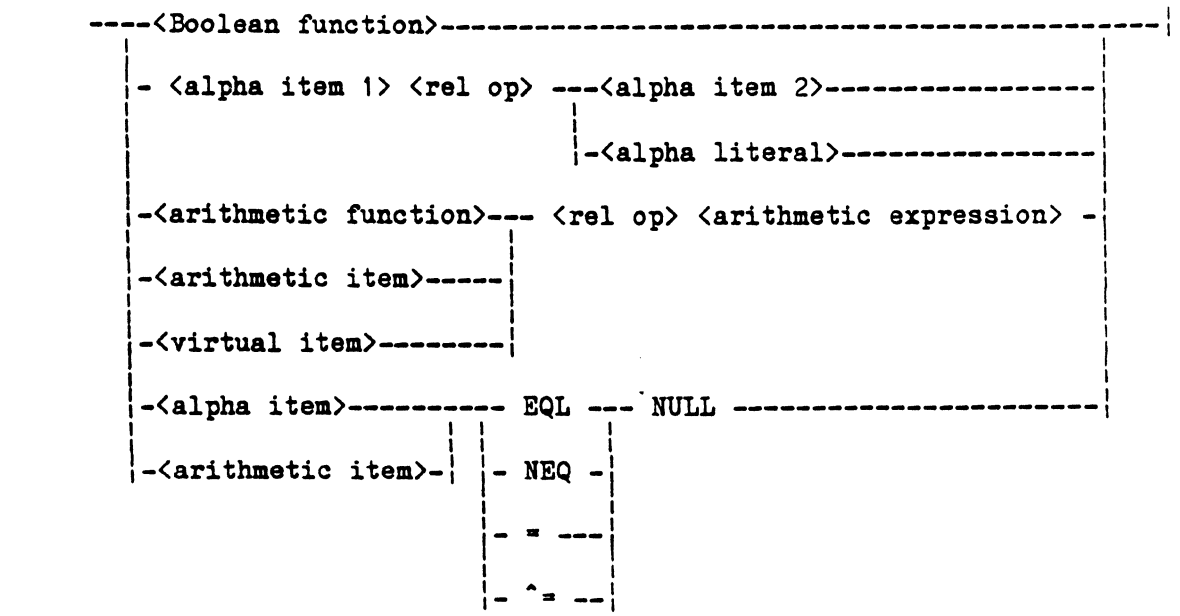
If a data item is undefined because the item is in a variable format that does not appear in a particular record, or because its subscript is undefined, both tests against NULL (EQL and NEQ) evaluate to FALSE.

Syntax:

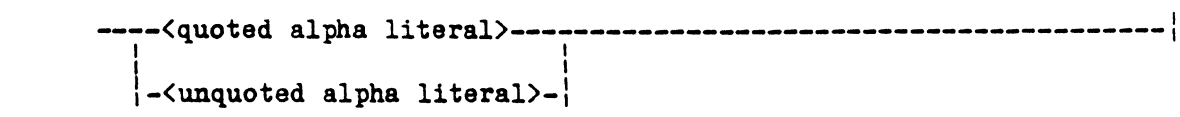
<selection condition>



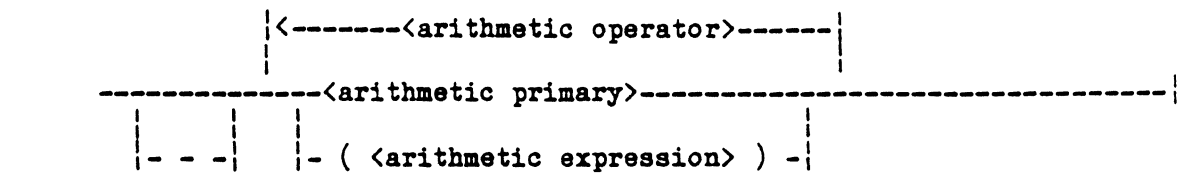
<Boolean primary>



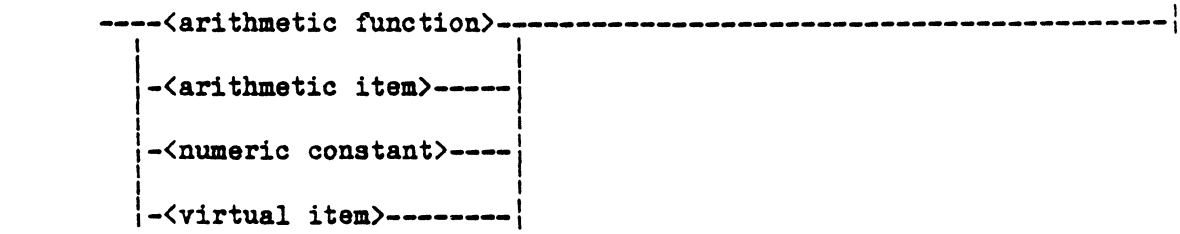
<alpha literal>



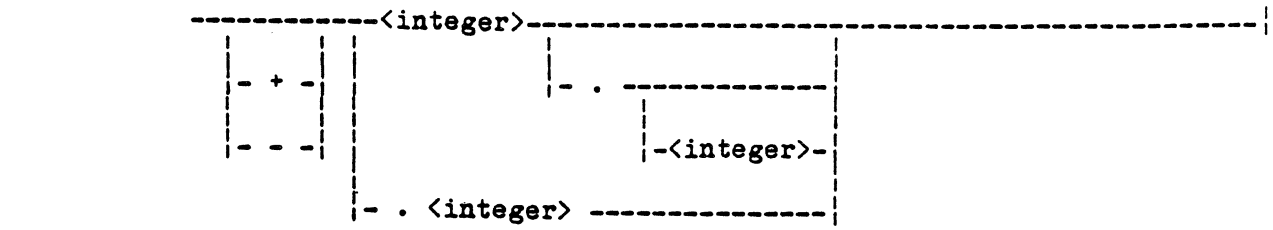
<arithmetic expression>



<arithmetic primary>



<numeric constant>



Semantics:

AND
OR
NOT

Boolean operators used to form complex selection conditions.

<Boolean item>

Item of type BOOLEAN in the selected record.

<Boolean primary>

Yields a value of TRUE or FALSE on evaluation.

<Boolean function>

Defined under <Boolean function> in this section. Must reference an item in an embedded structure within the record being selected.

<alpha item>
<alpha item 1>
<alpha item 2>

Items of type ALPHA. <alpha item 1> must be in the data set to be selected unless it is used in a <Boolean function>. <alpha item 2> cannot be in a data set embedded in the data set to be selected. If <alpha item 2> is not in the data set to be selected, then the data set containing <alpha item 2> must have a current record.

<alpha literal>

May be quoted or unquoted, depending on the command context and the value of OPTIONS QUOTES. (Refer to "OPTIONS Command" in this section.)

<quoted alpha literal>

A quoted sequence of characters containing uppercase characters A-Z, lowercase characters a-z, numeric characters 0-9, special characters, and blanks.

<unquoted alpha literal>

A sequence of characters containing only uppercase characters A-Z, lowercase characters a-z, and numeric characters 0-9. Special characters and embedded blanks are not allowed in an <unquoted alpha literal>.

<rel op>

Indicates that a relational operator is to be used. Relational operators are defined in the following table:

Operators	Symbolic Operators	Definition
EQL	=	Equal to.
NEQ	^=	Not equal to.
LSS	<	Less than.
LEQ	<=, ^>	Less than or equal to.
GTR	>	Greater than.
GEQ	>=, ^<	Greater than or equal to

<arithmetic function>

Defined under "<arithmetic function>" in this section.

<arithmetic item>

Item of type COUNT, FIELD, NUMBER, POPULATION, REAL, or RECORDTYPE in the record to be selected.

<virtual item>

Item established by the VIRTUAL command. When the value of the <virtual item> is computed, all data items it references must be in the current record or in records previously selected at a higher nesting level.

EQL NULL
NEQ NULL
= NULL
^= NULL

Tests the specified item for value of NULL.

<arithmetic primary>

Yields a numeric value on evaluation.

<arithmetic operator>

Defined in the following table:

Operator	Definition
+	Add.
-	Subtract (at least one blank must separate this operator from it operands, i.e., "A - B" is interpreted as A minus B. "A-B" is interpreted as data item "A-B
*	Multiply.
/	Divide.
DIV	Integer divide (yields an integer result and discards remainder).
MOD	Remainder divide (yields the remainder after an integer divide).

Examples:

The following selection condition specifies records in the data set EMP for which the Boolean item EMP-SEC-CLEAR is FALSE:

```
NOT EMP-SEC-CLEAR
```

The following selection condition specifies records in the data set EMP for which the item EMP-SALARY is greater than the average value of EMP-SALARY for all records in EMP:

```
EMP-SALARY GEQ AVG (EMP-SALARY)
```

The following selection condition specifies embedded EMP-ED records for which the virtual item OUR-GPA (refer to VIRTUAL ITEMS in Section 2) is greater than 7.5:

```
OUR-GPA > 7.5
```

The following selection condition specifies records in the master data set EMP for which all embedded EMP-ED records have a value other than "UNIV CAL DAVIS" for the item ED-INST-NAME:

```
ALL(ED-INST-NAME NEQ "UNIV CAL DAVIS")
```

<display list>

A <display list> specifies the data items or values to be displayed.

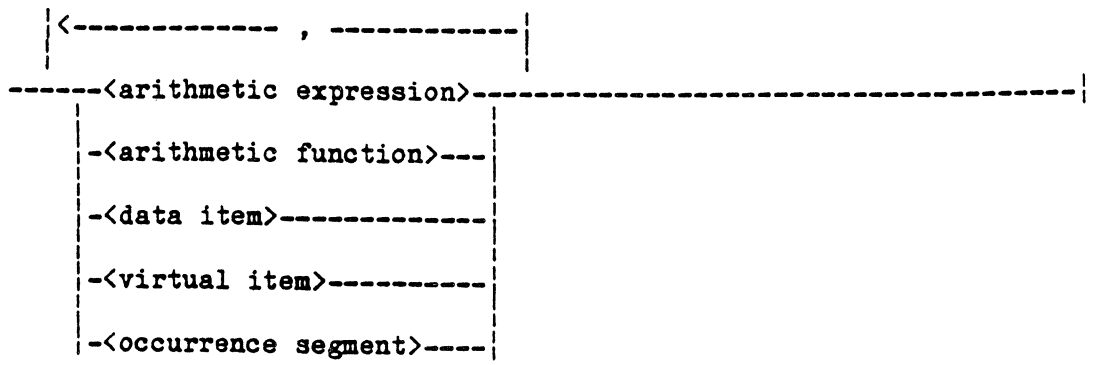
Qualifiers may be used to create unique names for data items that have the same name in different structures in the data base. Multiple levels of qualification may be used, if necessary.

If no subscripts are specified for an item, all occurrences of the specified item are displayed.

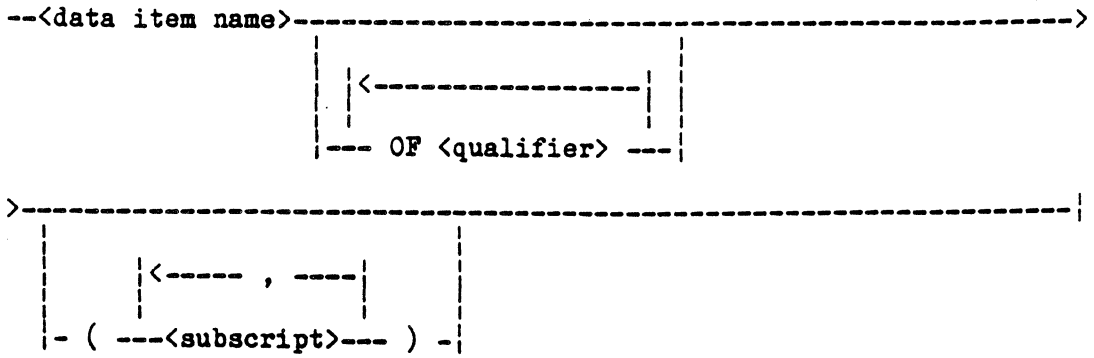
Particular data items need not be included in a <display list>. If desired, the result of an arithmetic expression or arithmetic current record function involving only numeric constants may be displayed. In this case, Inquiry is said to be in "calculator mode."

Syntax:

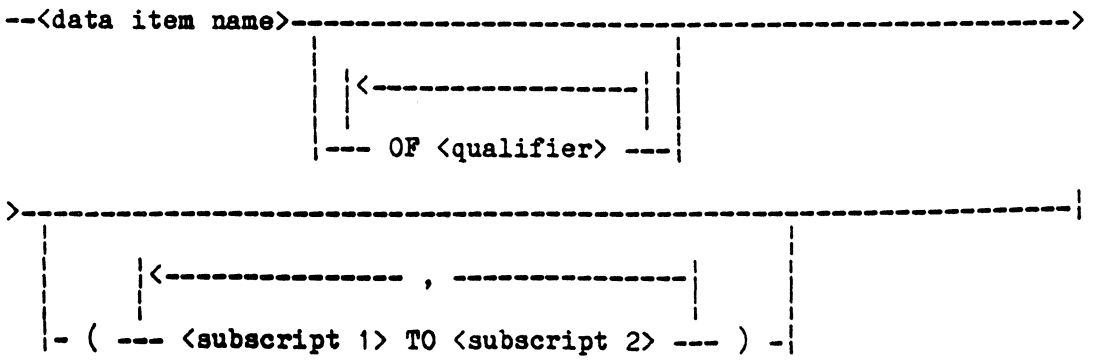
<display list>



<data item>



<occurrence segment>



<qualifier>

--<data set>----------

Semantics:

<arithmetic expression>

Defined under <selection condition> in "Common Syntactic Items."

<arithmetic function>

Defined under "Common Syntactic Items."

<virtual item>

Item established by the VIRTUAL command.

<data item>

Item to be displayed. This specification may require qualification or specification of a subscript to ensure uniqueness.

<occurrence segment>

Range of occurring elements that are displayed. All elements of the occurring item between <subscript 1> and <subscript 2>, inclusive, are displayed.

<data item name>

Name of a data item to be displayed. May be the name of a subscripted item or the name of an item present in more than one data set.

OF <qualifier>

Specifies the data set from which to display the data item, if data items in more than one data set have the same name.

<subscript>
<subscript 1>
<subscript 2>

Integer constant between 1 and the DASDL-declared limit for an item declared with an OCCURS clause in DASDL.

Examples:

The display list in the following DISPLAY command specifies that the result of the given arithmetic expression is to be displayed:

```
DISPLAY 4*(6000 MOD 4)/79655
```

The display list in the following DISPLAY command specifies that the following data items are to be displayed:

1. The result of the arithmetic function STD (standard deviation) over the item TYPE-SPEED for all appropriate selected records.
2. The items JOB-CODE and JOB-TITLE for the selected record or records.
3. The virtual item SKILLINDEX for the selected record or records.

```
DISPLAY STD(TYPE-SPEED), JOB-CODE, JOB-TITLE, SKILLINDEX
```

<sort option>

The <sort option> specifies the sequence in which records are accessed. A <sort option> specification can follow a selection condition in a SELECT or DISPLAY command and can be established, modified, or eliminated using the EDIT or SET commands.

If neither ASCENDING nor DESCENDING is specified, the access order is ASCENDING by default.

No more than 25 key items may be specified in a <sort option>. All key items that have been defined in the DASDL with an OCCURS clause must be subscripted using integer constants.

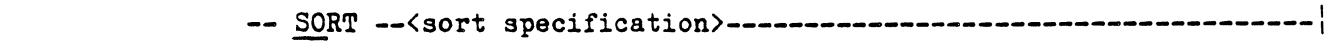
The <sort option> uses the system SORT intrinsic. During record selection, each record of the specified data set that satisfies the selection condition is read from the data base. An entry consisting of the extracted key items and the data base address of the record is made in a tag file; the tag file is then passed to the system SORT intrinsic, which sorts the tag entries in the sequence specified by the key item.

Inquiry reads the sorted tag file sequentially. For each tag entry, Inquiry reads the data set record at the address indicated in the tag entry. Inquiry then performs the same as if the data set record had been obtained directly from the data set. (For example, Inquiry displays items or executes the command associated with any attached embedded data set.)

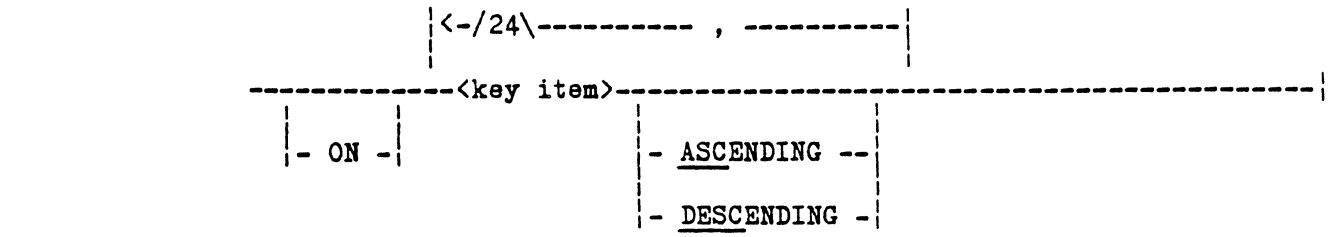
By default, the SORT intrinsic is allocated 12,000 words of core and 600,000 segments of disk. The SORT command can be used to alter these statements. (Refer to SORT Command in this section.)

Syntax:

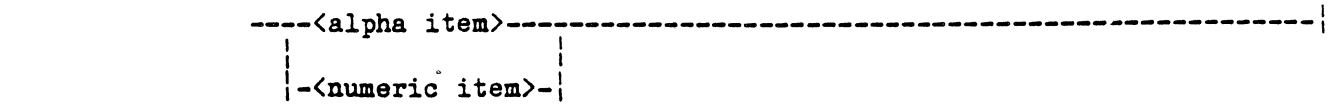
<sort option>



<sort specification>



<key item>



Semantics:

<sort specification>

Specifies key item on which to sort and the order of the sort.

ON

May be included to improve readability.

<key item>

Data item of the selected data set.

ASCENDING
DESCENDING

Specifies order of the sort.

<alpha item>

Item of type ALPHA.

<numeric item>

Item of type COUNT, FIELD, NUMBER, POPULATION, REAL, or RECORDTYPE.

Examples:

In the following example, the records in the data set EMP are sorted in ascending order of each employee's date of hiring (least recently hired to most recently hired):

```
-> DISPLAY EMP, SORT EMP-DATE-HIRED

1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:     8  MITTY        WALTER      963443636          540303
2:FALSE              2000.00
1:     6  MARLOWE      PHILLIP     867676865          551225
2:TRUE               2200.00
1:     3  JONSON       BEN         540022677          620115
2:TRUE               1771.56
1:     9  JOYCE        JAIME       364277866          650714
2:TRUE               2450.00
1:     2  HARDWICK     ALICE       647938292          701214
2:TRUE               3050.00
1:     5  CHAPMAN      GEORGE      166344267          710601
2:TRUE               2805.00
1:     4  DEKKER       THOMAS      554998666          711104
2:FALSE              1950.00
1:     1  FIELDS       JON         533733557          720312
2:TRUE               2250.00
1:     7  BURNS        GEORGINA    323313145          741010
2:FALSE              1750.00
```

In the following example, the records in EMP are sorted in descending order of each employee's date of hiring (most recently hired to least recently hired):

```
-> DISPLAY EMP, SORT EMP-DATE-HIRED DESCENDING

1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:     7  BURNS        GEORGINA    323313145          741010
2:FALSE              1750.00
1:     1  FIELDS       JON         533733557          720312
2:TRUE               2250.00
1:     4  DEKKER       THOMAS      554998666          711104
2:FALSE              1950.00
1:     5  CHAPMAN      GEORGE      166344267          710601
2:TRUE               2805.00
1:     2  HARDWICK     ALICE       647938292          701214
2:TRUE               3050.00
1:     9  JOYCE        JAIME       364277866          650714
2:TRUE               2450.00
1:     3  JONSON       BEN         540022677          620115
2:TRUE               1771.56
1:     6  MARLOWE      PHILLIP     867676865          551225
2:TRUE               2200.00
1:     8  MITTY        WALTER      963443636          540303
2:FALSE              2000.00
```

Functions

Inquiry supports the use of <Boolean function>s and <arithmetic function>s. These functions are described in the following paragraphs.

<Boolean function>

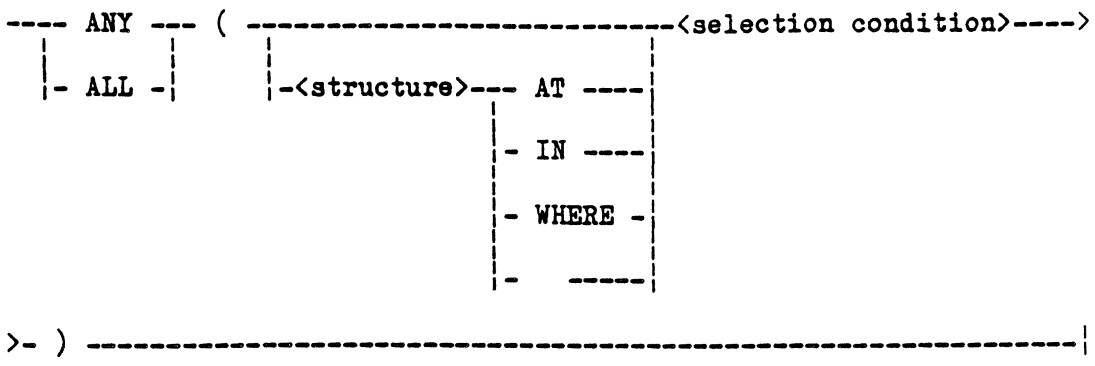
A <Boolean function> may be used to specify a selection condition involving an embedded structure.

If a selection condition specified within a <Boolean function> contains a relational operator, data items to the left of the relational operator must be declared in an embedded data set at the next lower nesting level. Data items to the right of the relational operator may be contained in the record to be selected, in the embedded data set, or in any record previously selected at a higher nesting level. If a relational operator is not included in the selection condition, the selection condition must reference data items declared in an embedded data set at the next lower nesting level. (Relational operators are defined under <selection condition> in "Common Syntactic Items.")

If a structure is specified in a <Boolean function>, only records retrieved through that structure are considered.

Syntax:

<Boolean function>



Semantics:

ANY

Yields the value TRUE if any record in the embedded data set satisfies the <selection condition>.

ALL

Yields the value TRUE if all records in the embedded data set satisfy the <selection condition>.

<structure>

An embedded structure in the selected record.

AT IN WHERE

Must separate embedded structure specification and <selection condition>.

<selection condition>

Defined under "Common Syntactic Items."

Example:

In the following example, the Boolean function returns a value of TRUE for records in ED-DATE-SET for which all occurrences of the item ED-GPA are greater than 2.0:

```
ALL (ED-GPA > 2.0)
```

In the following example, the Boolean function returns a value of TRUE for records in ED-DATE-SET for which any occurrence of the item ED-GPA is greater than 2.0:

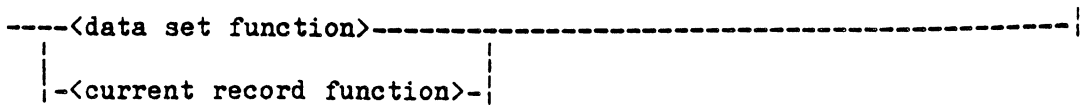
```
ANY (ED-GPA > 2.0)
```

<arithmetic Function>

An <arithmetic function> may be used to specify the result of an arithmetic operation on embedded or disjoint structures, numeric constants, or both. <arithmetic function>s are of two types: data set functions and current record functions. These types are discussed in the following paragraphs.

Syntax:

<arithmetic function>



DATA SET FUNCTIONS

Data set functions are functions of a sequence of values and may refer to several records. A number of records may be read and selected during the evaluation of these functions.

By default, all records in the specified structures are used in the computation. If a data set function includes a selection condition, only records that satisfy the selection condition are used in evaluating the function.

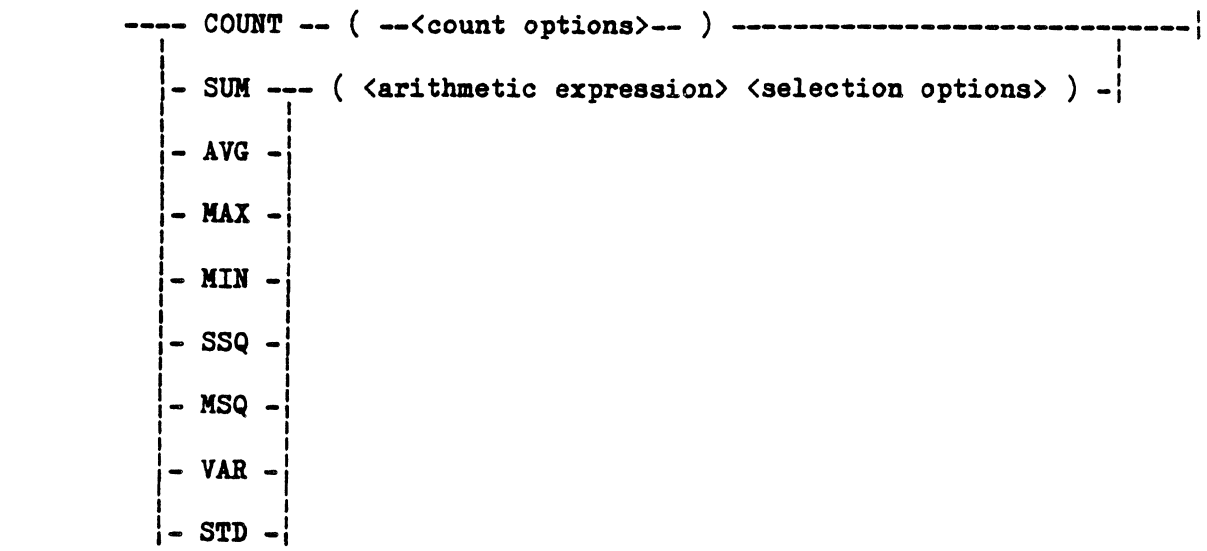
An <arithmetic function> is undefined if no records satisfy the specified selection condition. When an undefined situation is encountered, two hyphens are displayed instead of a value. Undefined items are described under UNDEFINED ITEMS in Section 2.

If a selection condition within a data set function contains a relational operator, data items to the right of the relational operator may be in the record to be selected or in any record previously selected at a higher embedding level. (Relational operators are defined under <selection condition> in COMMON SYNTACTIC ITEMS.)

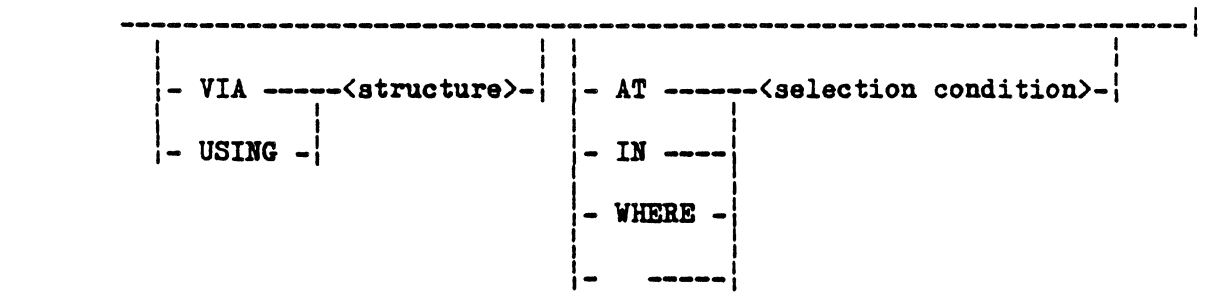
If a structure is designated in a data set function, only records retrieved through that structure are evaluated.

Syntax:

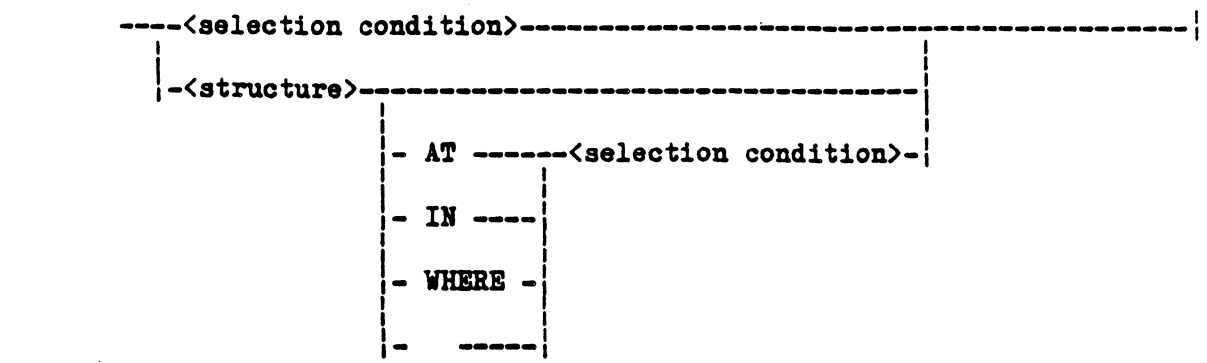
<data set functions>



<selection options>



<count options>



Semantics:

COUNT

Returns the number of records that satisfy the <selection condition> or the number of records in the structure.

<count options>

Establishes <selection condition>s or designates <structure>s for records input to the COUNT function.

<arithmetic expression>

Defined under <selection condition> in "Common Syntactic Items."

<selection options>

Establishes <selection condition>s or designates <structure>s for records input to <data set function>s except COUNT.

VIA
USING

Designates <structure> as the structure to be used to access records.

<structure>

The structure used to access records.

AT
IN
WHERE

Introduces <selection condition> for records to be used.

<selection condition>

Defined under "Common Syntactic Items."

SUM

Sum of the values of all items in the sequence.

AVG

Sum of the specified items in the sequence divided by the number of values summed.

MAX

Maximum value of the arithmetic item in the sequence.

MIN

Minimum value of the arithmetic item in the sequence.

SSQ

Sum of the squares of the specified items in the sequence.

MSQ

Mean square; sum of the squares of the specified items in the sequence divided by the number of items.

VAR

Variance; defined to be (SSQ-N*AVG**2)/(N-1), where N is the number of records selected.

STD

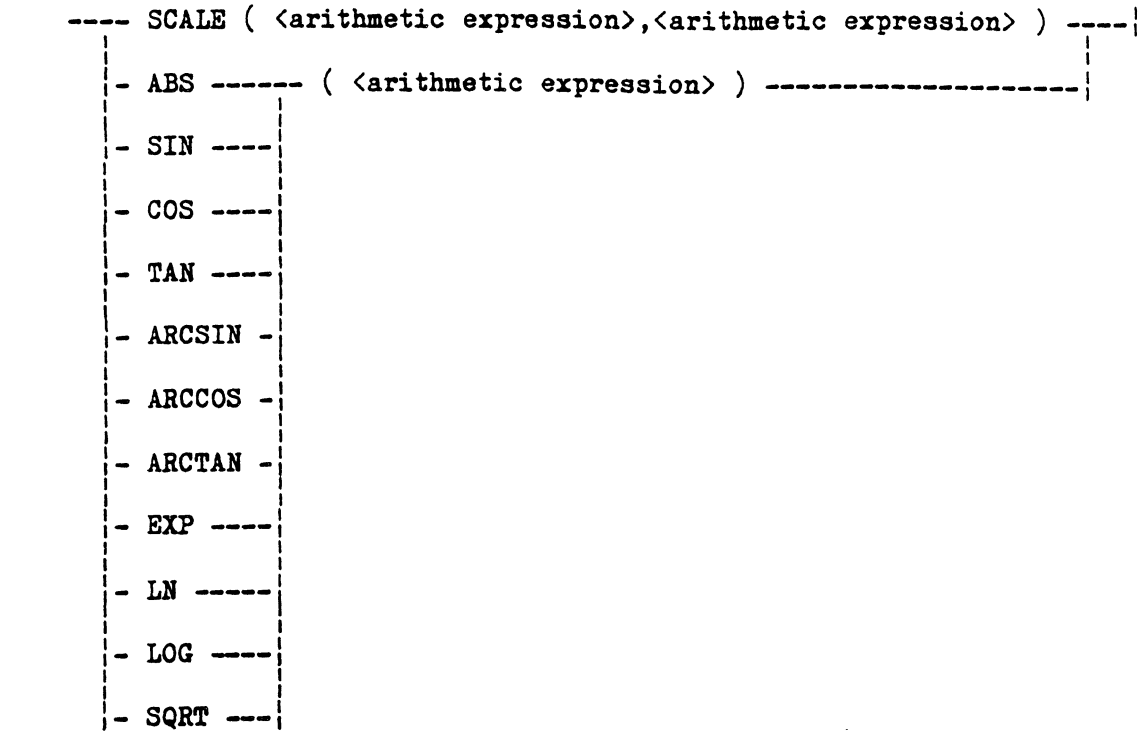
Standard deviation; square root of the variance.

CURRENT RECORD FUNCTIONS

A current record function is a function of a single value and must refer to a constant or an item in a currently selected record. No records are selected to evaluate a current record function.

Syntax:

<current record function>



Semantics:

In the descriptions below, X and F are <arithmetic expression>s. (<arithmetic expression> is defined under <selection condition> in "Common Syntactic Items.")

SCALE(X,F)

Returns the value of X rounded to F fractional digits. Non-fractional digits are unaffected.

ABS(X)

Absolute value of X.

SIN(X)

Sine of the angle X, where value of X is in radians.

COS(X)

Cosine of the angle X, where value of X is in radians.

TAN(X)

Tangent of the angle X, where value of X is in radians.

ARCSIN(X)

Inverse sine of X, where ABS(X) <= 1.

ARCCOS(X)

Inverse cosine of X, where ABS(X) <= 1.

ARCTAN(X)

Inverse tangent of X.

EXP(X)

Exponential function ("e" raised to the X power, where e = 2.718...).

LN(X)

Natural logarithm of X (base = e), where value of X > 0.

LOG(X)

Logarithm of X to base 10, where value of X > 0.

SQRT(X)

Square root of X, where value of X >= 0.

Examples:

The following arithmetic function returns the number of records in the data set JOB for which the item TYPE-SPEED is greater than 60:

```
COUNT(TYPE-SPEED > 60)
```

The following arithmetic function returns the number of records in the subset EMP-OVER10-SUBSET:

```
COUNT(EMP-OVER10-SUBSET)
```

The following arithmetic function returns the absolute value of the mean square of the values of EMP-SALARY in the subset EMP-OVER10-SUBSET:

```
ABS(MSQ(EMP-SALARY VIA EMP-OVER10-SUBSET))
```

INQUIRY COMMANDS

The following paragraphs give definitions, syntax, and semantics of commands used as input to Inquiry.

SELECT Command

The SELECT command locates a record that satisfies the selection specification contained in the command. If Inquiry locates a record that satisfies this specification, it displays a pound sign (#). If Inquiry fails to locate such a record, it displays the message #NONE.

The user is not required to specify the name of the data set to be selected. Inquiry can determine the data set from which to select records by examining the items specified in the SELECT command. If a structure is not specified, Inquiry uses sets when selecting records, if possible. If an appropriate set does not exist, Inquiry searches the data set directly.

Subsets, link items, and temporary subsets may not locate all records of the data set that satisfy the selection specification. Thus, Inquiry does not automatically use these structures, and they must be specified in the SELECT command if Inquiry is to use them to locate records.

Data sets must be selected in hierarchical sequence. To select records in an embedded data set, a record from the structure at the next higher nesting level must first be selected. All structures at a higher nesting level in the embedded family must have a current record selected.

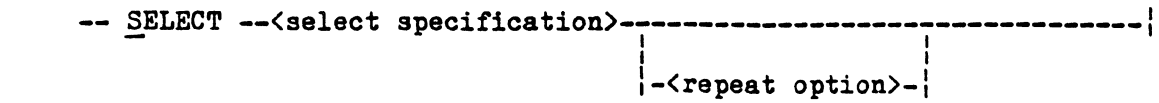
Records may be selected in a user-specified sequence. (Refer to <sort option> under "Common Syntactic Items" in this section.)

When items are selected using a link item or an embedded manual subset that refers to one of its own ancestors, the user may specify that only the data portion of the selected record is to be retrieved. In this case, the original sequence of records is retrieved. (By default, each link begins a new sequence.)

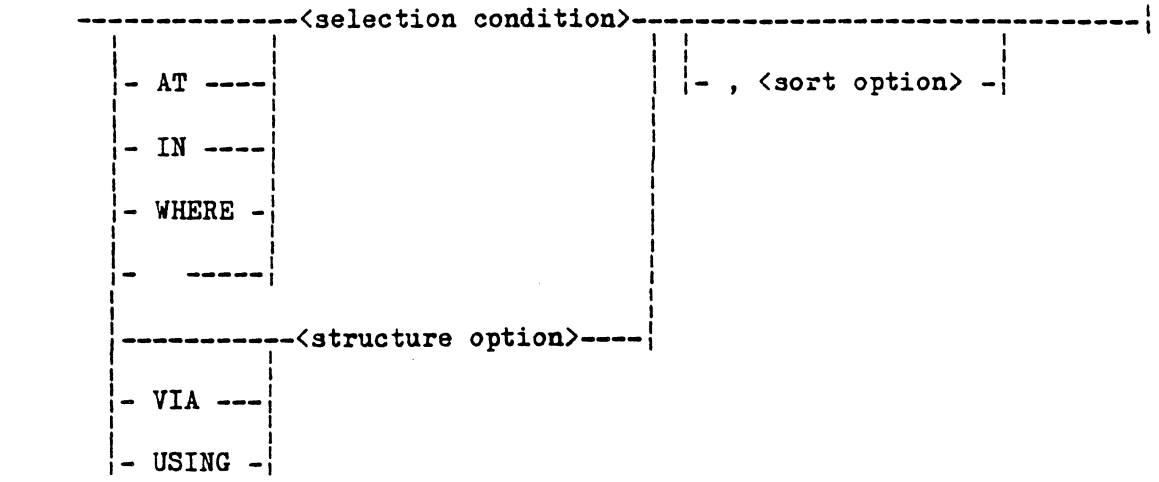
The SELECT command may contain a REPEAT option, which causes Inquiry to execute commands associated with the specified data set and its attached embedded structures. The REPEAT option is executed only if the

selection specifications preceding the option are satisfied.

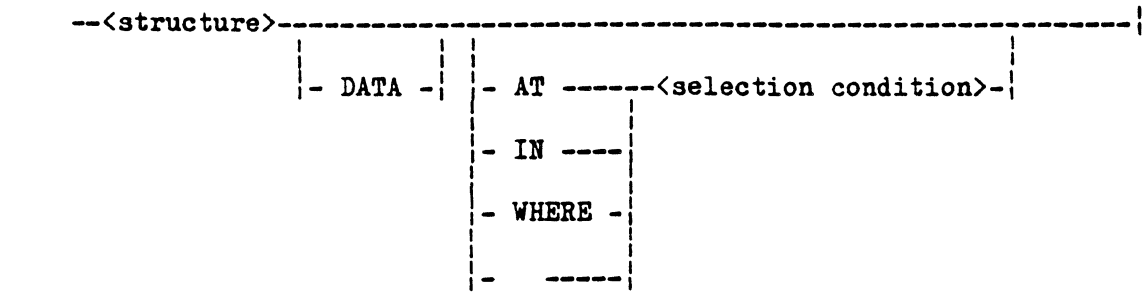
Syntax:



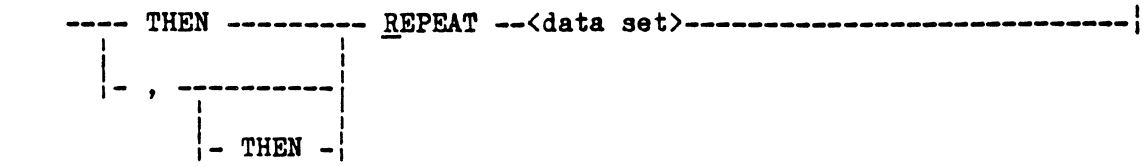
<select specification>



<structure option>



<repeat option>



Semantics:

<select specification>

Establishes <selection condition>s or designates <structure>s to use in record selection and specifies any sorting of selected records.

<repeat option>

Causes Inquiry to execute commands associated with <data set>. All <select specification>s must be satisfied before this option is executed.

<selection condition>

Defined under "Common Syntactic Items."

AT
IN
WHERE

Precedes specification of <selection condition>s.

<sort option>

Defined under "Common Syntactic Items."

VIA
USING

Precedes structure designation.

<structure option>

Specifies the <structure> to be used to access records.

<structure>

Data set, set, subset, link item, or a temporary subset created by the GENERATE command.

DATA

Retrieves only the data portion of the selected link item or embedded manual subset.

THEN

,

,THEN

Precedes REPEAT in <repeat option>.

REPEAT <data set>

Executes most recent SELECT or DISPLAY command associated with <data set>.

Examples:

The following SELECT command selects records from data set JOB for which JOB-TYPE is 2 and JOB-TITLE is not VICE PRESIDENT. (Inquiry is able to determine from the items specified which data set to use for selecting records.)

```
SELECT JOB-TYPE = 2 AND JOB-TITLE NEQ "VICE PRESIDENT"
```

The following SELECT command selects the same records as the commands in the preceding example, but selects them in alphabetical order of job titles:

```
SELECT JOB-TYPE = 2 AND JOB-TITLE NEQ "VICE PRESIDENT", SORT %
JOB-TITLE
```

The following SELECT command selects records in the subset EMP-OVER10-SUBSET for which EMP-DATE-HIRED is prior to the end of 1960, then executes the command associated with the data set ADR:

```
SELECT USING EMP-OVER10-SUBSET AT EMP-DATE-HIRED < 601231, %
THEN REPEAT ADR
```

For additional examples of SELECT commands, refer to "Selection Conditions" and "Structure Designation" in Section 2.

DISPLAY Command

The DISPLAY command displays data item names and their associated values or the results of arithmetic expressions and arithmetic current record functions. By default, all displayed output uses HEADING format unless a record displayed on a terminal screen is so large that its heading and values cannot fit on the screen. In this case, the output is displayed in TAB format. (Refer to Formatting Attributes in Section 2 for a discussion of these attributes.)

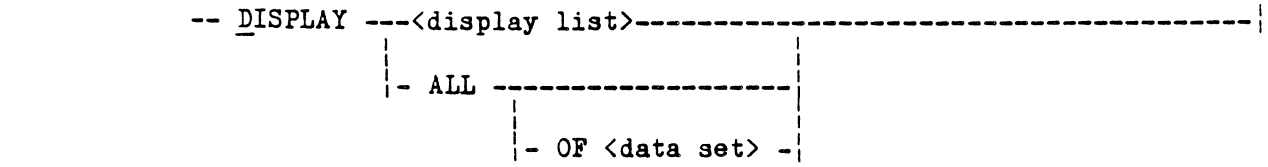
This command has two forms. Option 1 performs only display of items or values, and option 2 performs both record selection and item display. These forms are discussed in the following paragraphs.

Option 1:

This form of the DISPLAY command is used in one of two ways:

1. To display values from currently selected data set records.
2. To display the results of arithmetic expressions or arithmetic current record functions involving only numeric constants (calculator mode).

Syntax:



Semantics:

<display list>

Defined under "Common Syntactic Items" in this section. Either lists items to be displayed from selected records or specifies the <arithmetic expression> or <current record function> whose value is displayed.

ALL

Displays all items of the most recently selected record.

OF <data set>

Displays all items of the most recently selected record of this data set.

Examples:

In the following example, the Inquiry calculator mode is used to display the result of the given SIN function:

```
-> D SIN(476*5+(7 MOD 4)/9 + TAN(6))

                    SIN
  ------------------
      -0.95941046464
#
```

In all of the following examples, records of the data sets ADR and JOB have been selected in the order indicated using the following SELECT commands:

```
SELECT ADR

SELECT JOB AT JOB-TITLE = "VICE PRESIDENT"
```

In the first example, the items ADR-EMP-NO, and ADR-STREET-NO are displayed for the current ADR record:

```
-> D ADR-EMP-NO, ADR-STREET-NO

  ADR-EMP-NO  ADR-STREET-NO
  ----------  ----------------------------
           2  45 ALEXANDER RD DU BOIS PA
#
```

In the following example, all items from the last record selected (the record from JOB) are displayed:

```
-> DISPLAY ALL
```

```
1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
3:DRIVING-REQD  JOB-COMMENT
-------------------------------------------------------------------
1:        1          0      9999  VICE PRESIDENT
2:        --                  --  --            --
3:--             --
#
```

Option 2:

This form of the DISPLAY command combines item display and record selection. It may be used to display values from all records that meet a particular selection condition.

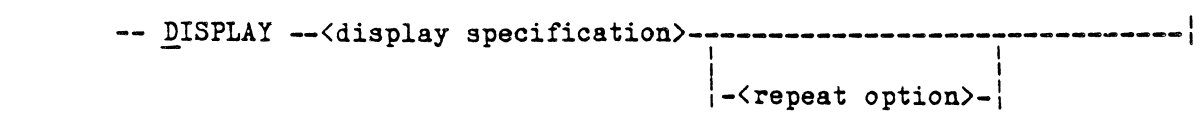
A limit may be included in this form of the DISPLAY command specifying the amount of uninterrupted output to be displayed. When the specified number of records have been displayed, Inquiry stops record selection and item display and waits for input from the user. If this specified number is greater than the number of records that can be displayed on a CRT, Inquiry displays as much information as possible and then waits for further input. If the user wishes to continue the item display at this point, the NEXT command or a blank input must be entered.

Records may be selected and displayed in a user-specified sequence. (Refer to <sort option> under "Common Syntactic Items" in this section.)

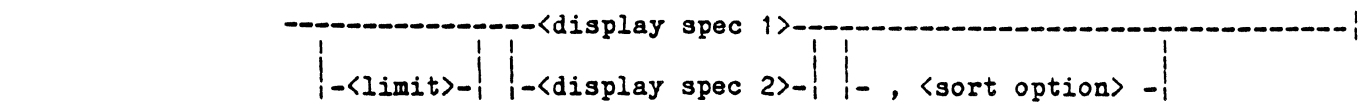
When items are selected using a link item or an embedded manual subset that refers to one of its own ancestors, the user may specify that only the data portion of the selected record is to be retrieved.

A REPEAT option may be specified with this form of the DISPLAY command. This option causes Inquiry to execute statements associated with the specified data set and its attached embedded structures. A REPEAT option is executed only when the selection specifications preceding the option in the DISPLAY command have been satisfied.

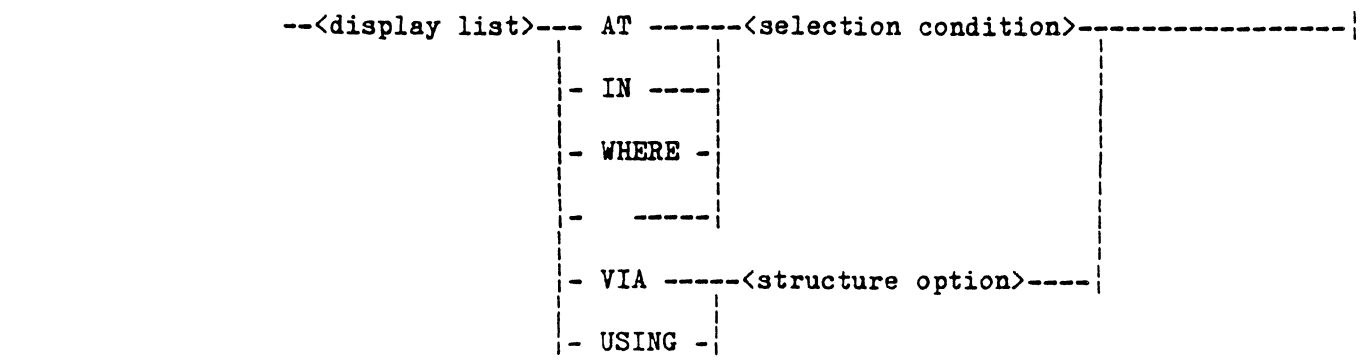
Syntax:



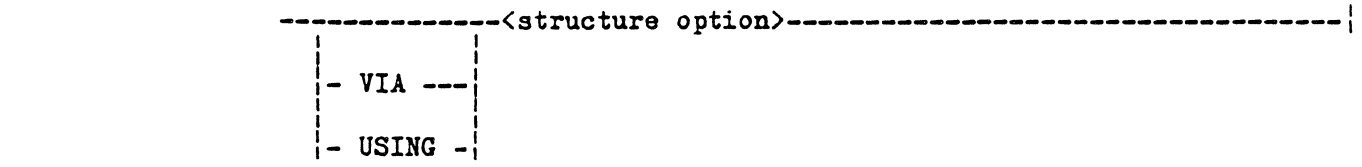
<display specification>



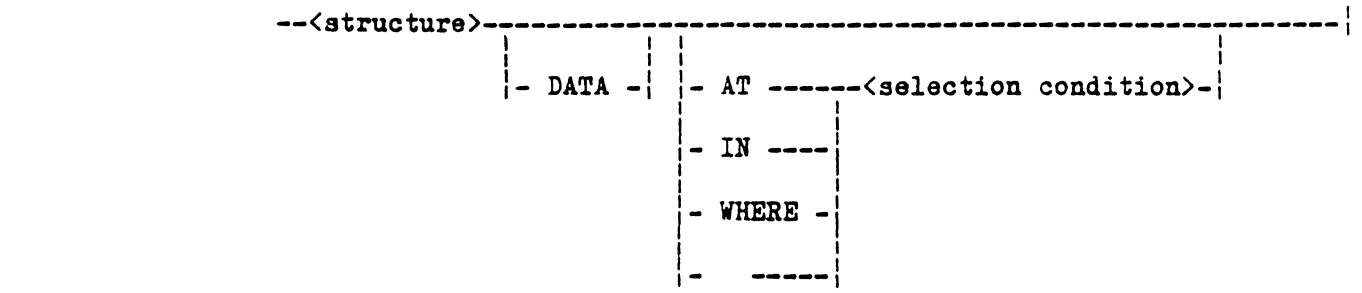
<display spec 1>



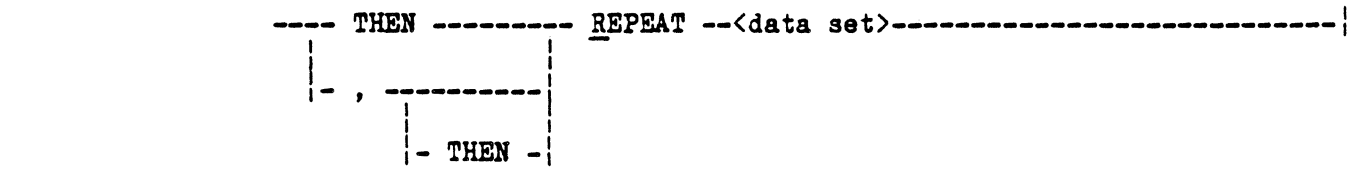
<display spec 2>



<structure option>



<repeat option>



Semantics:

<display specification>

Lists items, establishes <selection condition>s, or designates <structure>s for use in record display and specifies any sorting of displayed records.

<repeat option>

Establishes automatic looping between the selected data set and the data set following REPEAT. That is, after each display, the <select command> and <display command> associated with the <data set> specified in the <repeat option> (and any of its attached, embedded data sets or repeated data sets) are executed.

<limit>

An integer specifying the number of records to be displayed at a time.

<sort option>

Defined under "Common Syntactic Items."

<display list>

Records in this list meeting <display specification> are displayed.

AT
IN
WHERE

Precedes <selection condition> for records to be displayed.

<selection condition>

Defined under "Common Syntactic Items."

VIA
USING

Precedes <structure option>.

<structure option>

Displays records in the <structure> that satisfy the selection condition.

<structure>

Data set, set, subset, link item, or a temporary subset created by the GENERATE command.

DATA

Retrieves only the data portion of the selected link item or embedded manual subset.

THEN
,
,THEN

Must precede REPEAT in <repeat option>.

REPEAT <data set>

Executes most recent SELECT or DISPLAY command associated with <data set>.

Examples:

The following DISPLAY command displays records from the data set JOB for which JOB-TYPE is 2 and JOB-TITLE is not "VICE PRESIDENT". (Inquiry is able to determine from the items specified which data set to use for displaying records.)

```
-> DISPLAY AT JOB-TYPE = 2 AND JOB-TITLE NEQ "VICE PRESIDENT"

   1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
   2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
   3:DRIVING-REQD  JOB-COMMENT
   ------------------------------------------------------------------
   1:        1         2      3333  SALESMAN
   2:        --                 --  --           TRUE
   3:FALSE         MINUMUM 2 YRS FIELD EXP
   1:        1         2      7777  SALES MANAGER
   2:        --                 --  --           TRUE
   3:FALSE         5 YRS SALES EXP PLUS DEGREE
   1:        2         2      5000  PROJECT LEADER
   2:        --                 --  --           FALSE
   3:FALSE         3 YRS MANAGEMENT EXP PLUS DEGREE
   #
```

The following DISPLAY command displays the same records as the commands in the preceding example, but displays them in alphabetical order of job titles:

```
-> DISPLAY AT JOB-TYPE = 2 AND JOB-TITLE NEQ "VICE PRESIDENT", SORT %
   #%
-> JOB-TITLE

   1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
   2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
   3:DRIVING-REQD  JOB-COMMENT
   ------------------------------------------------------------------
   1:        2         2      5000  PROJECT LEADER
   2:        --                 --  --           FALSE
   3:FALSE         3 YRS MANAGEMENT EXP PLUS DEGREE
   1:        1         2      7777  SALES MANAGER
   2:        --                 --  --           TRUE
   3:FALSE         5 YRS SALES EXP PLUS DEGREE
   1:        1         2      3333  SALESMAN
   2:        --                 --  --           TRUE
   3:FALSE         MINUMUM 2 YRS FIELD EXP
   #
```

In this example, a SET command is used to associate the DISPLAY command with the data set ADR (refer to SET Command in this section):

```
SET ADR TO DISPLAY ADR-STREET-NO,ADR-ZIP AT ADR-EMP-NO = EMP-NO
```

The following DISPLAY command displays records in the subset EMP-OVER10-SUBSET for which EMP-DATE-HIRED is prior to December 12, 1960, then executes the command associated with the data set ADR for each record displayed from EMP-OVER10-SUBSET:

```
-> DISPLAY USING EMP-OVER10-SUBSET AT EMP-DATE-HIRED < 601212, THEN %
   #%
-> REPEAT ADR

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     6  MARLOWE     PHILLIP     867676865          551225
   2:TRUE              2200.00
     ADR-STREET-NO                   ADR-ZIP
     ------------------------------  -------
     23 PHILIPS ST APT 2 MARLENA GA    42323
   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     8  MITTY       WALTER      963443636          540303
   2:FALSE             2000.00
     ADR-STREET-NO                   ADR-ZIP
     ------------------------------  -------
     19 23RD ST NEW YORK NY            20867
   #
```

SELECT/DISPLAY Command

If Inquiry is used on a data base containing embedded structures, the SELECT command and option 2 of the DISPLAY command can be combined in a single input command to display records from a structure embedded in the selected structure. Inquiry processes the combined SELECT/DISPLAY command as a series of nested loops.

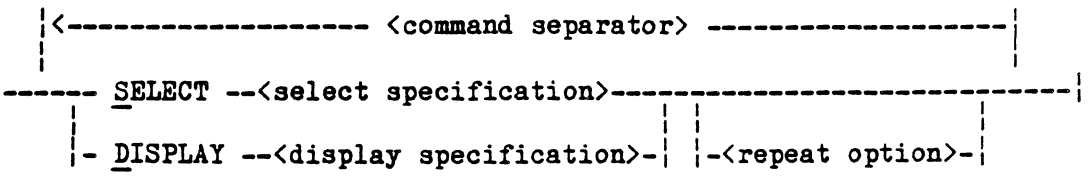
Structures must be selected in hierarchical sequence throughout the family of embedded structures. Inquiry processes each command within this command for the first record that satisfies the selection criteria.

Once a record is selected at the lowest valid nesting level, if a select specification is input for that level, Inquiry responds with a pound sign (#) and waits for further input. However, if a display specification is input, Inquiry processes all records in that structure that satisfy the display specification.

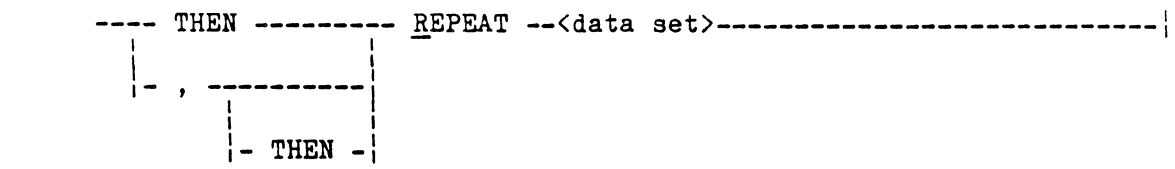
When all records at this nesting level are exhausted, Inquiry attempts to retrieve a new record at the next higher nesting level. If a display specification is input, Inquiry automatically retrieves the next record in that structure that satisfies the display specification and continues to process the original input command from that point. If all records at this nesting level are exhausted, Inquiry proceeds to the next higher nesting level. This process continues until either all records are exhausted or Inquiry encounters a select specification.

A REPEAT option can be included in a SELECT/DISPLAY command to invoke automatic looping outside the normal family of embedded structures. Automatic looping is invoked when the selection criteria within the family of embedded structures associated with REPEAT have been satisfied. A previous SELECT command, or an option-2 DISPLAY command, must already have been specified for the REPEATed data set (or associated with the repeated data set by using a SET command).

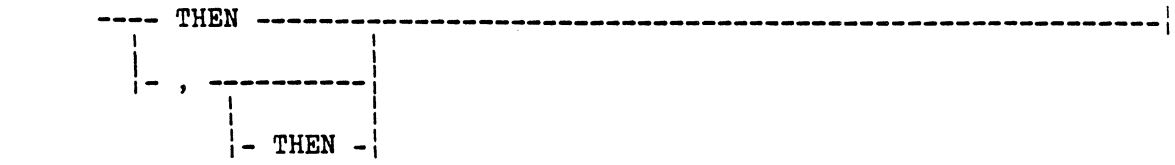
Syntax:



<repeat option>



<command separator>



Semantics:

<select specification>

Establishes <selection condition>s or designates <structure>s for use in record selection and specifies any sorting of selected records. (Refer to SELECT Command in this section for complete syntax and semantics.)

<display specification>

Lists items, establishes <selection condition>s, or designates <structure>s for use in record display and specifies any sorting of displayed records. (Refer to DISPLAY Command in this section for complete syntax and semantics.)

<repeat option>

Causes commands associated with the <data set> following REPEAT to be executed.

THEN

,

,THEN

Precedes REPEAT in <repeat option> or separates SELECT and DISPLAY commands in the combined command. THEN preceded by a comma is required if the following SELECT or DISPLAY command applies to an embedded structure.

REPEAT <data set>

Executes most recent SELECT or DISPLAY command associated with <data set>.

Examples:

The following combined SELECT/DISPLAY command selects a record from EMP, displays the items ED-INST-NAME, ED-COURSE, and ED-GPA for every school at which the employee had a grade-point average better than 3.0, and executes the command for the data set ADR ("DISPLAY ADR-STREET-NO,ADR-ZIP AT ADR-EMP-NO = EMP-NO") that has already been established by a SET command:

```
-> SELECT EMP,THEN DISPLAY ED-INST-NAME, ED-COURSE,ED-GPA AT ED-GPA > %

  #%

-> 3, THEN REPEAT ADR

  ED-INST-NAME                  ED-COURSE   ED-GPA
  ----------------------------  ----------  ------
  KENT STATE                    PSYCHOLOGY    3.77
  ADR-STREET-NO                 ADR-ZIP
  ----------------------------  -------
  5 MAIN ST NEWTOWN PA            56662
  ED-INST-NAME                  ED-COURSE   ED-GPA
  ----------------------------  ----------  ------
  UCLA                          BUS ADM       4.00
  ADR-STREET-NO                 ADR-ZIP
  ----------------------------  -------
  5 MAIN ST NEWTOWN PA            56662
 #
```

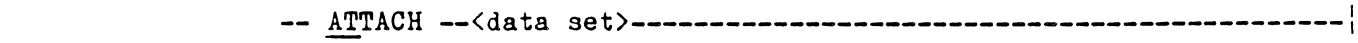
ATTACH Command

Inquiry remembers the most recently entered command for each data set. The ATTACH command is used to combine the command entered for an embedded data set with the command entered for the master data set, thus allowing automatic looping between the two structures. When the ATTACH command has been entered, the command associated with the embedded data set is executed each time the command associated with the master data set is executed. The combined command is executed using the REPEAT command.

To attach an embedded data set to its master data set, the master data set must have a current record.

All structures selected in a combined SELECT/DISPLAY command are attached by default.

Syntax:



Semantics:

<data set>

An embedded data set.

Example:

In the following example, the buffers for the master data set EMP and for the embedded data set EMP-ED are set to the specified DISPLAY commands, and then automatic looping between the two structures is established using the ATTACH command:

```
-> SET EMP TO DISPLAY 1 EMP

   #

-> SET EMP-ED TO DISPLAY ED-DATE-SET

   #
```

```
-> ATTACH EMP-ED

   #

-> REPEAT EMP

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     1  FIELDS      JON         533733557          720312
   2:TRUE              2250.00
   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 610101  UNIV OF MARYLAND                          5266  SOC SCI
   2:  2.89
   1: 650615  KENT STATE                                2123  PSYCHOLOGY
   2:  3.77
   #

-> NEXT EMP

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     2  HARDWICK    ALICE       647938292          701214
   2:TRUE              3050.00
   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 450615  UNIV OF SO CAL                            2125  HISTORY
   2:  4.00
   1: 650615  UCLA                                      2238  MBA
   2:  4.00
   #
```

CLEAR Command

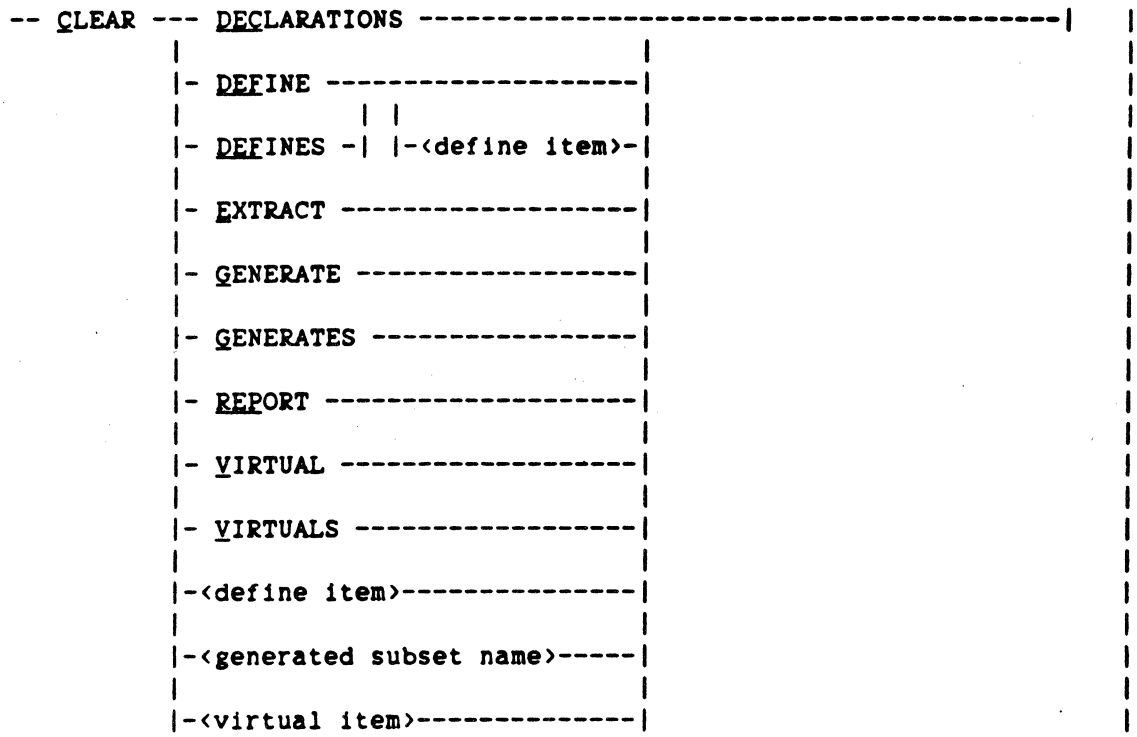
The CLEAR command may be used to discard define items, virtual items, and generated subsets. An item that has been cleared may no longer be referenced; however, such an item may be re-established by:

1. Using the RESTORE command (if the declaration had previously been saved).

2. Recreating the item in the normal manner.

The CLEAR command does not remove any declarations from the definitions file. However, if a SAVE command is entered after a CLEAR command, or at the end of the Inquiry session, the declarations that have been cleared are removed from the definitions file. If SAVE is not entered, the cleared declarations remain available during concurrent and subsequent Inquiry sessions.

Syntax:



Semantics:

DECLARATIONS

Deletes all define, virtual, and generate items.

DEFINE
DEFINES
GENERATE
GENERATES
VIRTUAL
VIRTUALS

Deletes all items of the specified type.

DEFINE <define item>
DEFINES <define item>
<define item>
<generated subset name>
<virtual item>

Deletes individual items of the specified type.

EXTRACT

Deletes all information associated with a previously entered **EXTRACT** command and resets extract variables to their initial values and extract file attributes to their default values.

REPORT

Returns all internal storage used to remember the report statements and generate the report; may be entered to improve efficiency or to avoid possible conflict between a previous report definition and a new report generation.

Examples:

In the following example, the CLEAR command is used to remove all define items from the data base:

```
-> SHOW DEFINES

   DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE AND
    EMP-DATE-HIRED < LDATE
   DEFINE SHOWZIP = D ADR-ZIP
   #

-> CLEAR DEFINES

   #

-> SHOW DEFINES

   #
```

In the following example, the CLEAR command is used to remove the generated subset PROTECTORS:

```
-> SHOW GENERATES
```

```
   GENERATE RICHEMP = EMP-SALARY > 2000
   GENERATE OLDTIMERS = EMP-DATE-HIRED < 600101
   GENERATE TRUSTIES = RICHEMP AND OLDTIMERS
   GENERATE CRONIES = RICHEMP OR OLDTIMERS
   GENERATE PROTECTORS = RICHEMP + OLDTIMERS
   GENERATE DEFENDERS = RICHEMP - OLDTIMERS
   #

-> CLEAR PROTECTORS

   #

-> SHOW GENERATES

   GENERATE RICHEMP = EMP-SALARY > 2000
   GENERATE OLDTIMERS = EMP-DATE-HIRED < 600101
   GENERATE TRUSTIES = RICHEMP AND OLDTIMERS
   GENERATE CRONIES = RICHEMP OR OLDTIMERS
   GENERATE DEFENDERS = RICHEMP - OLDTIMERS
   #
```

CREATE Command

The CREATE command creates a new record to be added to a data set.

All items specified for the new record in a CREATE command must belong to the same data set. In addition, any variables used in the specification of arithmetic items must reference values in currently selected records of other data sets. The following types of items cannot be specified in a CREATE command:

1. Count items.
2. Link items.
3. Population items.
4. Record-type items.
5. Items declared as READONLY.

Items of the record to be created that are not assigned values in the CREATE command are initialized to one of the following values:

1. DASDL-declared INITIALVALUE, if present.
2. DASDL-declared NULL, if present.
3. Default NULL.

To override these defaults, NULL may be specified in the CREATE command for all data items except required items and items such as Boolean items that cannot take a null value. (Refer to the DASDL Reference Manual for definitions of these terms.)

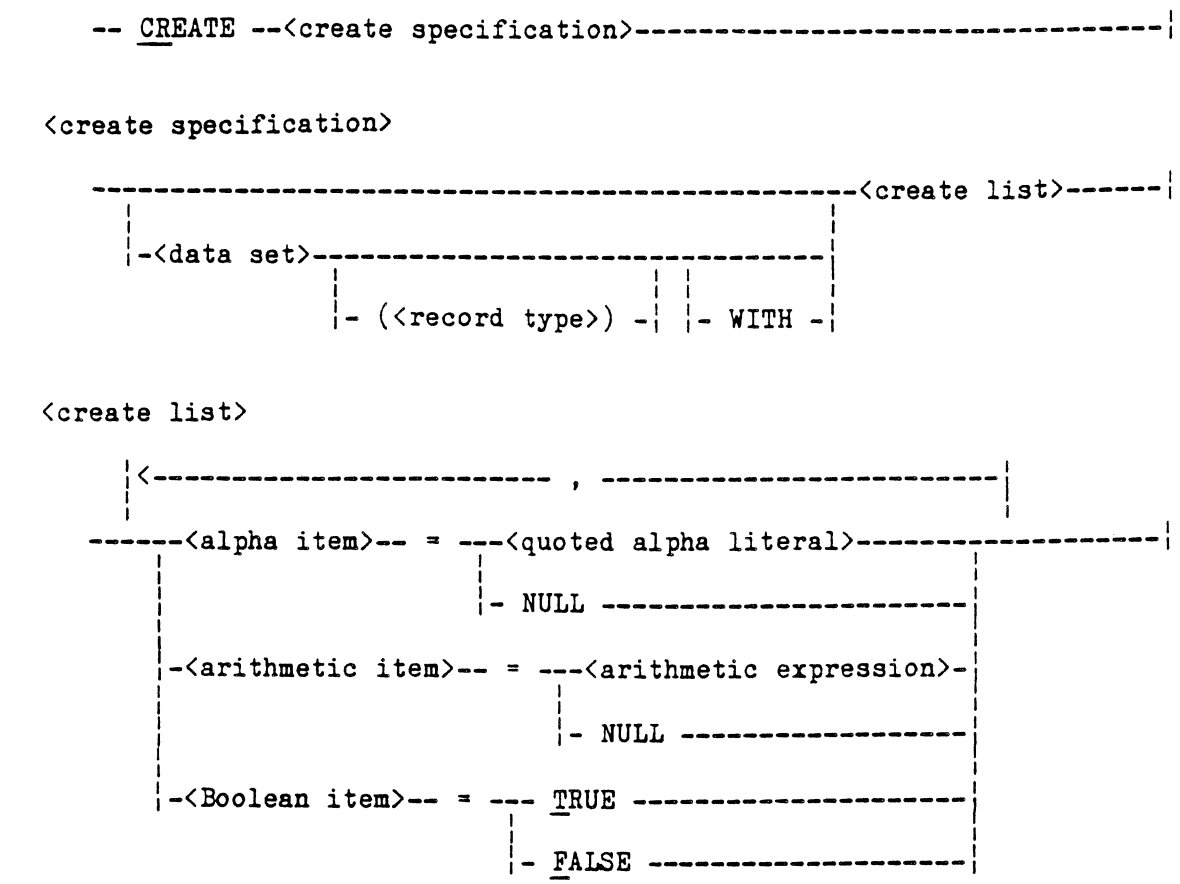
A CREATE causes any existing selection of the data set itself, or any selection of any of its embedded data sets, to be discontinued. A successful CREATE operation causes the newly created record to become the selected record.

To create records in an embedded data set, a record of the corresponding master data set must be currently selected.

The CREATE command invokes the data base ACCESSROUTINES. The Inquiry CREATE command is equivalent to a user-language CREATE statement followed by a STORE statement, and any violation of the normal DMSII restrictions related to the create and store operations causes an error condition. (Refer to the User Language Interface SOG for details regarding these statements and restrictions.) If such an error condition is detected, an error message is displayed, and the record is not created.

CREATE operations performed on an audited data base are audited in the usual DMSII manner. (Refer to Modification of Audited Data Bases in Section 2 of this manual.)

Syntax:



Semantics:

<create specification>

List of items and their values in the newly created record. This list may include a specification of the data set that will contain the created record, the record type of the created record, or both.

<data set>

Explicitly names the data set for which the record is to be created. If qualification is required (that is, if all items in the <create list> appear in more than one data set), either <data set> must be specified or the items must be qualified in the usual way. (Refer

to "Qualification" in Section 2.) Otherwise, Inquiry can determine the proper data set from the items referenced in the <create list>.

<record type>

A positive <integer> specifying the record type for a data set with variable-format records. Must appear when creating a record for a data set with variable-format records and must be one of the valid record types specified in the DASDL for the data base.

WITH

An optional keyword that may be used to improve readability.

<create list>

Specifies items and their values in the new record.

<alpha item>
<arithmetic item>
<Boolean item>

Items in an existing data set that are assigned values in the newly created record.

<quoted alpha literal>

Defined in the semantics under <selection condition> in COMMON SYNTACTIC ITEMS. Must be enclosed in quotes even if OPTIONS QUOTES = FALSE. (Refer to OPTIONS Command in this section.)

<arithmetic expression>

Yields the value assigned to <arithmetic item> in the newly created record. Variables used in this expression must reference values in currently selected records of another data set.

Examples:

In the following example, a new record is created in the data set EMP for an employee named Candy Stack:

```
-> CREATE EMP-NO = 11, EMP-LNAME = "STACK", EMP-FNAME = "CANDY",%
#%
```

```
 -> EMP-SSN = 573230911, EMP-DATE-HIRED = 811223, EMP-SEC-CLEAR = TRUE,%
    #%
 -> EMP-SALARY = 4500

    #RECORD OF EMP CREATED
    #
```

The newly created record is then displayed:

```
 -> D EMP AT EMP-NO = 11

    1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
    2:EMP-SEC-CLEAR  EMP-SALARY
    ------------------------------------------------------------------
    1:    11  STACK       CANDY       573230911          811223
    2:TRUE              4500.00
    #
```

In the following example, a new record is created in the JOB data set for a custodial engineer job:

```
 -> CREATE JOB (2) JOB-CODE = 3737, JOB-TITLE = "CUSTOD ENGNR",%
    #%
 -> SPANISH-REQD = FALSE, DRIVING-REQD = FALSE, JOB-COMMENT =%
    #%
 -> "REQUIRES NO SKILLS"

    RECORD OF JOB CREATED
    #

 -> D JOB AT JOB-CODE = 3737

    1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
    2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
    3:DRIVING-REQD  JOB-COMMENT
    ------------------------------------------------------------------
    1:        0         2      3737  CUSTOD ENGNR
    2:        --                 --  --           FALSE
    3:FALSE         REQUIRES NO SKILLS
    #
```

DEFINE Command

The DEFINE command establishes a name that can be used as a shorthand notation for any Inquiry text. This text can be an entire Inquiry command, several Inquiry commands, or any part of an Inquiry command.

A define name cannot be the name of an Inquiry command, data base item, or generated subset.

A multiple-command define allows more than one statement to be submitted for immediate execution. Such a define either must end with a semicolon or must be invoked last in an input or multiple-command define. A multiple-command define must not contain DEFINE, EDIT, or REPEAT-with-EDIT commands.

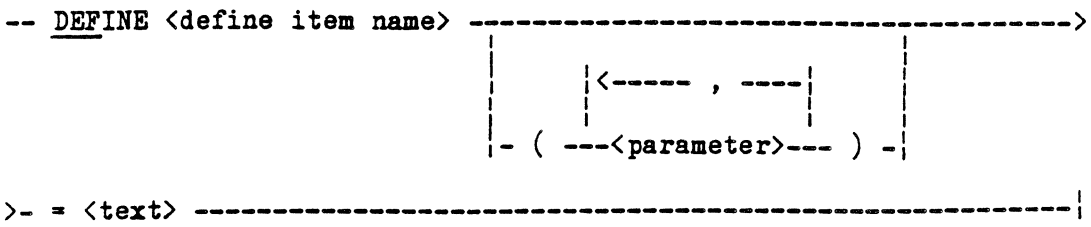
In addition, a multiple-command define must contain complete commands. Other defines may be invoked; however, any command started in a multiple-command define cannot be completed in another multiple-command define.

A multiple-command define may be viewed as a subroutine containing complete statements; that is, an invocation of a multiple-command define is itself a command, the same as a subroutine call. However, like any define, a multiple-command define is not parsed and verified until it is invoked. Therefore, syntax errors and violations of the rules governing the use of multiple-command defines are not detected until the define is used.

If an error occurs during the execution of any command contained in a multiple-command define at any level of invocation, all remaining commands in the define are ignored, and the system waits for further input.

A parametric define allows parameters to be supplied at the time the define item is referenced. When a parametric define is referenced, the formal parameters appearing in the text of the define are replaced by the actual parameters.

Syntax:



Semantics:

<define item name>

A unique identifier. Cannot be an Inquiry command, data base item identifier, or generated subset name.

<text>

Any legal Inquiry input text. For a multiple-command define, <text> cannot include DEFINE, EDIT, or REPEAT-with-EDIT commands. Each command in a multiple-command define must be separated (and may be followed) by semicolons.

<parameter>

Establishes a variable to be used in define text.

Examples:

In the following example, the define item NEWHIRE is created to display the records of employees hired after January 1, 1970:

```
-> DEFINE NEWHIRE = DISPLAY EMP AT EMP-DATE-HIRED > 700101

   #
```

```
-> NEWHIRE

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     1  FIELDS      JON          533733557          720312
   2:FALSE             2000.00
   1:     2  KRANTZ      ALICE        647938292          701214
   2:TRUE              3050.00
   1:     4  DEKKER      THOMAS       554998666          711104
   2:FALSE             1950.00
   1:     5  CHAPMAN     GEORGE       166344267          710601
   2:TRUE              2805.00
   1:     7  BURNS       GEORGINA     323313145          741010
   2:FALSE             1750.00
   1:    11  STACK       CANDY        573230911          811223
   2:TRUE              4500.00
   #
```

In the following example, the parametric define HIRERANGE is created to display the records in EMP for employees hired between any two specified dates:

```
-> DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE %
   #%
-> AND EMP-DATE-HIRED < LDATE

   #

-> HIRERANGE (650101,700101)

   12EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     9  JOYCE       JAIME        364277866          650714
   2:TRUE              2450.00
   #
```

DELETE Command

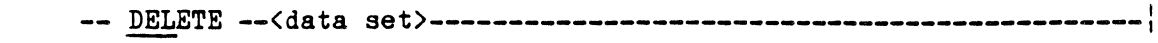
The DELETE command removes the currently selected record from a data set. After this command is executed, items from the deleted record may still be displayed until a new record for that data set is selected.

If a record has not been selected for the data set, an error message is displayed and deletion does not occur.

DELETE invokes the data base ACCESSROUTINES. Any violation of the normal DMSII restrictions related to the DELETE operation causes an error condition, and an error message is displayed. If an error condition is detected, deletion does not occur.

Deletions made to an audited data base are audited in the usual DMSII manner. (Refer to Modification of Audited Data Bases in Section 2 of this manual.)

Syntax:



Semantics:

<data set>

Deletes currently selected record for this data set.

Examples:

In the following example, a specified record of the data set JOB is selected and then deleted:

```
-> SELECT JOB AT JOB-CODE = 3000

   #
```

```
-> DELETE JOB

   SELECTED RECORD OF JOB DELETED
   #
```

Because the deleted record remains the current record, items can be displayed from it:

```
-> DISPLAY JOB-CODE, JOB-TITLE

   JOB-CODE  JOB-TITLE
   --------  ----------------------------
       3000  EXEC SECRETARY
   #

-> DISPLAY JOB AT JOB-CODE = 3000

   #NONE
   #
```

For additional examples, refer to "CREATE" and DELETE in Section 3.

DETACH Command

The DETACH command separates an embedded data set from its master data set and prevents automatic looping between the two data sets.

If the master data set and embedded data set are specified in the same SELECT or option-2 DISPLAY command, they are attached by default.

When the master data set is selected, the embedded data set is marked as having no current record. Inquiry displays an appropriate message if an attempt is made to display any items contained in the embedded data set.

Syntax:

```
-- DETACH --<data set>-----------------------------------------|
```

Semantics:

<data set>

Separates this embedded data set from its master data set.

Example:

The embedded data set EMP-ED is attached to the master data set EMP using an option-2 DISPLAY command:

```
-> SELECT EMP, THEN DISPLAY EMP-LNAME,EMP-FNAME,ED-GPA AT ED-COURSE=%
   %#
-> "SOC SCI"
   EMP-LNAME    EMP-FNAME    ED-GPA
   ----------   ----------   ------
   FIELDS       JON            2.89
   #
```

Next, the DETACH command is used to detach EMP-ED from EMP:

```
-> DETACH EMP-ED

   #
```

Because EMP-ED is detached from EMP, automatic looping between the two structures is prevented. Thus, the following REPEAT command selects the first EMP record:

```
-> REPEAT EMP

   #
```

Because a record has been selected from EMP, EMP-ED has no current record, and no items from EMP-ED can be displayed:

```
-> DISPLAY EMP-LNAME,EMP-FNAME

   EMP-LNAME    EMP-FNAME
   ----------   ----------
   FIELDS       JON
   #

-> DISPLAY ED-GPA

   ED-GPA
   ------
   #OWNER OF ED-GPA (EMP-ED) NOT SELECTED
   #
```

EDIT Command

The EDIT command modifies the text in the current buffer.

The current buffer normally contains the text of the most recently entered DEFINE, DISPLAY, GENERATE, SELECT, SET, or VIRTUAL command. The RECALL command may be used to replace the contents of the current buffer by the text of a previously entered command. (Refer to RECALL Command in this section.)

After each EDIT command is entered, the contents of the current buffer are displayed. EDIT may be performed as many times as necessary to alter the contents.

To edit only the text of a specific line, the line number of the desired line must be entered prior to the first delimiter. The line number must correspond to one of the lines displayed; if not, a syntax error results. The line numbers are not saved; they merely indicate the position of the line within the entire text. If any subsequent changes to the text occur (for example, statement insertions or deletions), the line numbers of succeeding lines also change. Consequently, when future EDIT operations take place, new text and new line numbers are displayed.

EDIT does not REPEAT or reprocess the modified text; therefore, a sequence of modifications may be lost if a REPEAT command is not entered before other input. REPEAT-with-EDIT can be used to cause the new text to be processed immediately. (Refer to REPEAT Command in this section.)

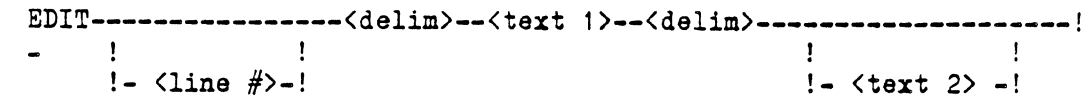
An EDIT command immediately followed by a save command is not enough to save the edited version of a declaration. It is necessary to either:

1. Repeat the edited declaration before saving it (EDIT, REPEAT, SAVE).

or

2. Use the REPEAT command instead of the EDIT command to do the editing (REPEAT, SAVE).

Syntax:



Semantics:

<text 1>
<text 2>

If <text 2> is specified, replaces the first occurrence of <text 1> by <text 2>; otherwise, eliminates <text 1> from the current buffer.

<delim>

Any special character not contained in <text 1>. Both <delim>s must be the same character.

<line #>

Edits only this line in text.

Examples:

In the following example, the specifications for the generated subset RICHEMP are to be edited. These specifications are reloaded using the RECALL command as follows:

```
-> RECALL RICHEMP

   GENERATE RICHEMP = EMP-SALARY > 2000
   #
```

Next, the EDIT command is used to change the specifications for RICHEMP:

```
-> EDIT /2000/4000

   GENERATE RICHEMP = EMP-SALARY > 4000
   #
```

The REPEAT command is then used to save the changed specifications, and the generated subset that satisfies the changed specifications is displayed:

```
-> REPEAT

   #WAIT
   #

-> DISPLAY RICHEMP

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:    11  STACK       CANDY       573230911          811223
   2:TRUE              4500.00
   #
```

In the following example, the VIRTUAL command that created the virtual item EMP-YEAR-SALARY is reloaded into the current buffer and edited. The REPEAT command causes Inquiry to remember the new specification for the virtual item. The edited virtual item is then displayed for the first record in EMP:

```
-> RECALL EMP-YEAR-SALARY

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12
   #

-> EDIT /EMP-SALARY * 12/EMP-SALARY * 12 + 2500

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12 + 2500
   #

-> REPEAT

   #


-> S EMP

   #

-> D EMP-LNAME, EMP-SALARY, EMP-YEAR-SALARY

   EMP-LNAME   EMP-SALARY     EMP-YEAR-SALARY
   ----------  ----------  ------------------
   FIELDS         5000.00               62500
   #
```

In the following example, one of the four lines of text in the current buffer is edited:

```
-> SHOW

   01:SET JOB TO SELECT JOB AT JOB-TYPE = 2;
   02:SET JOB TO SORT JOB-TYPE;
   03:SET JOB TO REPEAT EMP;
   04:SET EMP TO DISPLAY JOB-EMP-SUBSET
   #

-> EDIT 02 /JOB-TYPE/JOB-CODE

   01:SET JOB TO SELECT JOB AT JOB-TYPE = 2;
   02:SET JOB TO SORT JOB-CODE;
   03:SET JOB TO REPEAT EMP;
   04:SET EMP TO DISPLAY JOB-EMP-SUBSET
   #
```

EXTRACT Command

The EXTRACT command names items in the selected record to be extracted to a file and specifies the format of these items in the extract file. The GENERATE command initiates the actual extraction process; an extract file and a report can be generated using a single GENERATE command.

The following types of items may be specified in an EXTRACT command:

1. Data base items, including count, population, record-type, and group items. Link items may not be specified. If an item is declared with an OCCURS clause, the subscript must be provided; this subscript may be an arithmetic expression.

2. Virtual items.

3. Inquiry functions.

4. Arithmetic expressions.

Items may be renamed in the record layout summary. If a new name is not specified, data base items and virtual items keep their original names, and functions and arithmetic expressions are assigned names by Inquiry.

Extracted items may be reformatted. Table 4-1 indicates the allowable format specifications and the default formats used for the extract file.

Table 4-1. Item Formats in Extract Files

Source Item	Extract File Format			
	ALPHA	FIELD	NUMBER	REAL
Alpha	Default	No	No	No
Boolean	No	Yes	Default	Yes
Field	No	Default	Yes	Yes
Number	No	Yes	Default	Yes
Real	No	Yes	Yes	Default
Inquiry Func.	No	Yes	Yes	Default
Arith. Exp.	No	Yes	Yes	Default
Population	No	Default	Yes	Yes
Record-Type	No	Default	Yes	Yes
Count	No	Default	Yes	Yes
Group	Default	No	No	No

The following paragraphs discuss the formats listed in Table 4-1.

ALPHA

This format resizes an alpha source item. The specified item size for this format may be no larger than 4095. If the specified item size is smaller than the source-item size, the left-most characters of the source item appear in the extract file and the right-most characters are truncated.

FIELD

Items in this format appear as unsigned, binary integers. Boolean source items are converted to field items with binary values of 0 or 1. Number and real source items, including Inquiry functions and arithmetic expressions, are converted to unsigned, binary integers. The specified item size for this format must be less than 48; this value is always rounded to the nearest 4-bit multiple.

NUMBER

This format extracts source items as signed or unsigned, fractional or integer values. Number items may add or delete a sign or change or delete a scale factor when extracted in this format. Boolean, field, and real items are converted to a number item with the

specified precision and, optionally, the specified scale factor. The maximum specified precision is 23 for unsigned items and 22 for signed items; if specified, the scale factor may not exceed the specified precision.

REAL

This format extracts source items as signed or unsigned, fractional or integer values. Values are represented in single-precision, floating-point form. The size is 1 word (12 4-bit digits). Boolean, field, and number items may be extracted in this format. The maximum specified precision for unsigned items is 12 and for signed items is 11. If specified, the scale factor may not exceed the specified precision.

By default, source items are extracted in their original format. Arithmetic expressions and Inquiry functions are extracted in REAL format. Boolean items are extracted as 1-digit, unsigned values in NUMBER format. Count, field, and population items are extracted in FIELD format; a field item is treated as an unsigned, binary integer regardless of whether the field item contains Boolean items. If individual bits are extracted, the Boolean bit names must be used. By default, each Boolean bit is then extracted as a 1-digit value in NUMBER format; however, the bit could be reformatted in the same way as a Boolean source item.

The EXTRACT command may specify the starting position for items in the extract record. The first position in the extract record is zero. Alpha and real source items must begin on byte boundaries (that is, the specified starting position must be an even number). Specified items must be listed in order of increasing position; a COLUMN OVERLAP error occurs if the position specified for an item is a position before the ending of the previous item. Unused space between items in the extract file is described in the record-layout summary as FILLER with a 4-bit zero value for the appropriate number of digits.

Error conditions that may be caused during extract procedures are listed below along with the resulting default values for items in the extract file.

1. Null errors. These errors occur when:

   a. The data base item being extracted is null. If such an error occurs, the default value for the item in the extract file is the data base null value.

   b. The item does not exist in the current variable-format record. The default extract value is 4-bit zeros.

   c. The item references a data set other than the data set specified in the GENERATE <data set> command, and no

current record is selected for that data set. The default extract value is 4-bit zeros.

2. Conversion errors. These errors occur when:

   a. The value to be placed in FIELD format is less than 0 or too large to fit in the field.

   b. An item is extracted in unsigned NUMBER format and the value of the item is less than 0.

   c. An item is extracted in NUMBER format and the value of the item is greater than the largest value (or, if the item is negative, less than the smallest value) that can be stored in the nonfractional precision of the extract NUMBER format.

   Default extract values from conversion errors are 4-bit zeros for all items.

3. Fault errors. These errors occur when:

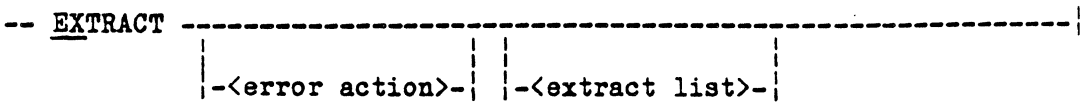
   a. A fault is detected during the evaluation of a virtual item, arithmetic expression, or Inquiry function. (Division by zero is an example of such a fault.)

   b. The subscript of an item defined with an OCCURS clause produces an invalid offset.

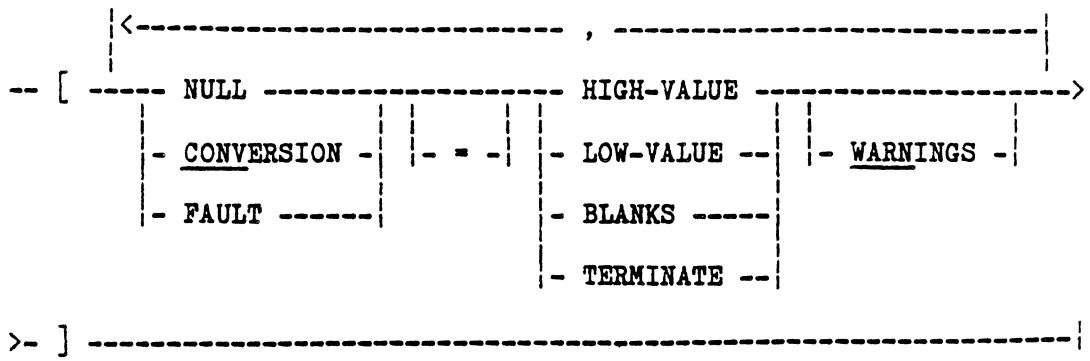
   Default extract values from fault errors are 4-bit zeros for all items.

The EXTRACT command can specify error actions to be taken for each extracted item or for all extracted items. (In the event of conflicting specifications, the specification for an individual item takes precedence.) In addition, the EXTRACT command can specify extract values, which override the default extract values in the event of an error.

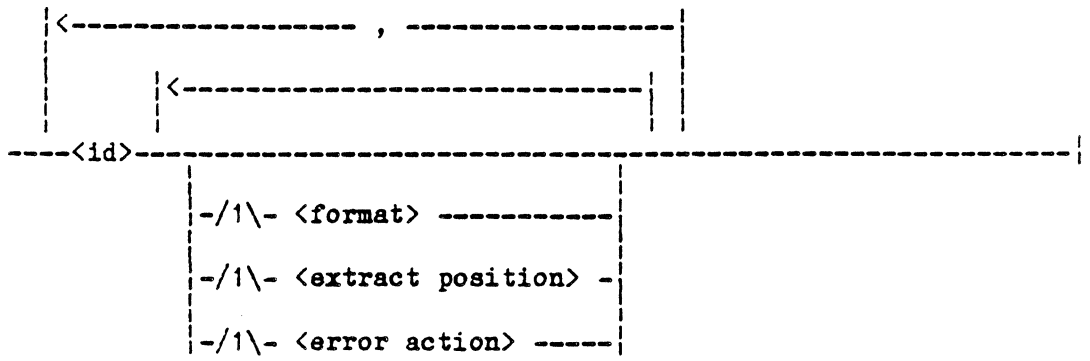
Syntax:



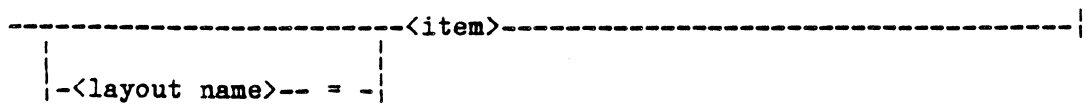
<error action>



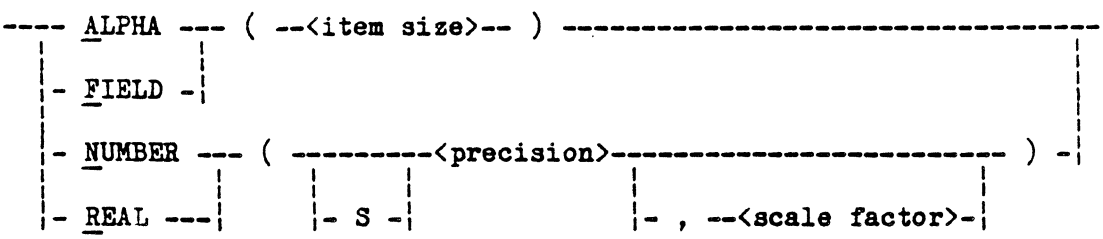
<extract list>



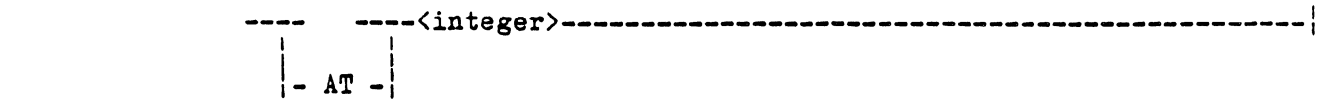
<id>



<format>



<extract position>



Semantics:

EXTRACT

If no other options are specified, displays or prints record-layout summary for previously entered EXTRACT command.

<error action>

Specifies action to be taken in the event of extract errors.

<extract list>

Specifies items to be extracted, position of each item in the extract record, and action to be taken in the event of extract errors.

NULL
CONVERSION
FAULT

Specify actions in the event the specified extract error occurs.

HIGH-VALUE
LOW-VALUE

In the event of specified error(s), value of extracted items is all bits ON or OFF, respectively.

BLANKS

In the event of specified error(s), value of extracted items is all blanks.

TERMINATE

In the event of specified error(s), extracts remaining fields in the current extract record, terminates the extract process, and generates a WARNINGS message. After termination, the current data set record is accessible to other Inquiry commands.

WARNINGS

In the event of specified error(s), generates a message containing the extract key, type of error, and item in error. This option is RESET by default.

<format>

Specifies reformatting for extracted items.

<extract position>

Specifies starting position for extracted item in the extract record.

<id>

Specifies item name in record layout summary.

<layout name>

An alphanumeric identifier of 17 or fewer characters beginning with a letter.

ALPHA
FIELD
NUMBER
REAL

Specifies format of item in extract file.

(<item size>)

Specifies size of item extracted in ALPHA or FIELD format.

(<precision>)

Specifies precision of an item extracted in NUMBER or REAL format.

S

Used for signed items extracted in NUMBER or REAL format.

<scale factor>

Must not exceed <precision>.

AT

Precedes <extract position> specification.

Examples:

The EXTRACT command in this example makes the following specifications for the extract file:

1. The item JOB-CODE is extracted at an offset of 2.

2. The item JOB-TITLE is extracted at an offset of 10.

3. The arithmetic expression TYPE-SPEED + SHORTHAND-SPEED / 100 is extracted under the name SKILLINDEX at an offset of 72. If the expression has a value of null, the resulting value in the file is to contain all blanks.

```
-> EXTRACT JOB-CODE   2, JOB-TITLE   10, SKILLINDEX = TYPE-SPEED + %
   #%
-> SHORTHAND-SPEED/100 [NULL=BLANKS] AT 72

   # EXTRACT RECORD SIZE WILL BE 7 WORDS UNLESS CHANGED BY FILE COMMAND
   # EXTRACT FILE LAYOUT (EXTRACT/DIV68 ON DISK.)
     ITEM              TYPE          OFFSET
   FILLER            NUMBER(2)          0
   JOB-CODE          NUMBER(4)          2
   FILLER            NUMBER(4)          6
   JOB-TITLE         ALPHA(30)         10
   FILLER            NUMBER(2)         70
   SKILLINDEX        REAL              72
   #
```

For an additional example, refer to "Inquiry EXTRACT" in this section.
3.

FILE Command

The FILE command displays or sets commonly used file attributes for an extract file. The assigned values must be valid for the specified attributes (refer to the attribute descriptions in the I/O Subsystem Reference Manual for valid values).

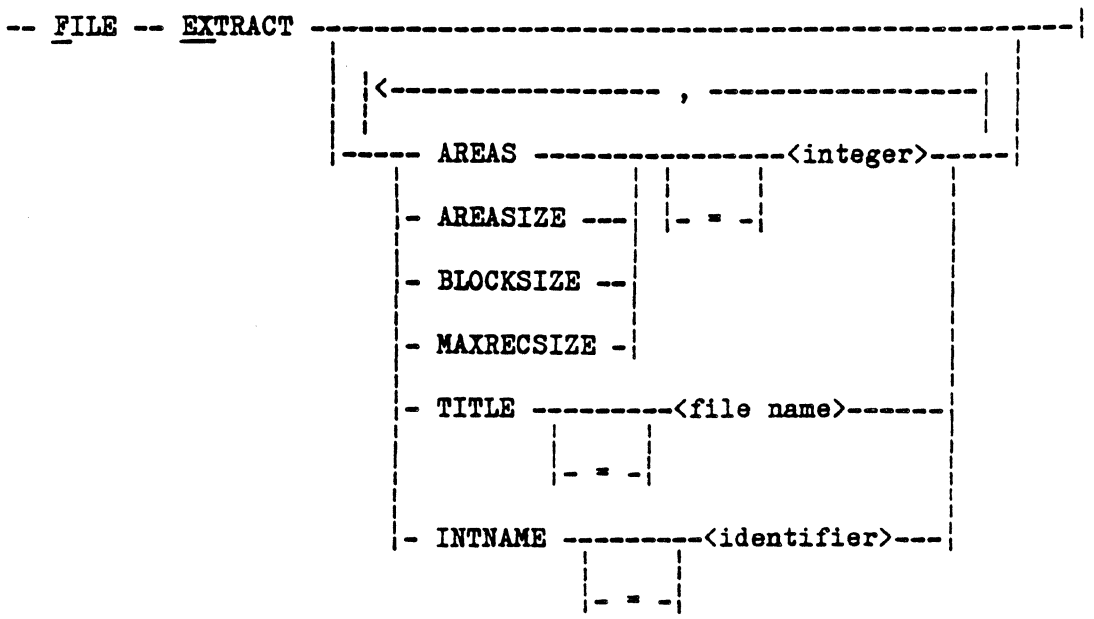
The default attribute values are 20 for AREAS, 1000 for AREASIZE, 30 for BLOCKSIZE, and EXTRACT/<data base name> for TITLE. (The CLEAR EXTRACT command resets extract file attributes to their default values.)

Normally, the FILE command is entered after the EXTRACT command has determined the default MAXRECSIZE. If the MAXRECSIZE specified in the FILE command is smaller than the default value, the extract record is truncated. If the MAXRECSIZE specified in the FILE command is larger than the default value, the extract record is padded with 4-bit zeros.

If BLOCKSIZE is not either 0 or an even multiple of MAXRECSIZE, a warning message is issued, and BLOCKSIZE is set to 0 in order to avoid a run-time error.

If a value is specified for the INTNAME attribute, this attribute is set so that file-equated attributes in a WFL statement can be attached to the extract file.

Syntax:



Semantics:

FILE EXTRACT

If no other options are specified, displays values of extract file attributes.

AREAS
AREASIZE
BLOCKSIZE
MAXRECSIZE
TITLE
INTNAME

Assigns values to specified attribute(s).

=

May be included for documentation.

<file name>

Name specified to override the default extract file name.

<identifier>

Value assigned to the INTNAME attribute of the extract file.

Examples:

The following FILE command displays the default values for Inquiry extract file attributes:

```
-> FILE EXTRACT

     TITLE = EXTRACT/DIV68 ON DISK.
     INTNAME = EXTRACT.
     MAXRECSIZE = 30
     BLOCKSIZE = 30
     AREAS = 20
     AREASIZE = 1000
  #
```

The following FILE command displays the extract file attribute values for the file created in the example under EXTRACT Command in this section:

```
-> FILE EXTRACT

     TITLE = EXTRACT/DIV68 ON DISK.
     INTNAME = EXTRACT.
     MAXRECSIZE = 3
     BLOCKSIZE = 30
     AREAS = 20
     AREASIZE = 1000
  #
```

The following FILE command changes the value of the MAXRECSIZE attribute to 2, the value of the AREASIZE attribute to 500, and the value of the TITLE attribute to "EXT/DIV68" for the file in the preceding example:

```
-> FILE EXTRACT MAXRECSIZE=2,AREASIZE=500,TITLE=EXT/DIV68

  #
```

```
-> FILE EXTRACT

      TITLE = EXT/DIV68 ON DISK.
      INTNAME = EXTRACT.
      MAXRECSIZE = 2
      BLOCKSIZE = 30
      AREAS = 20
      AREASIZE = 500
   #
```

GENERATE Command

The GENERATE command can be used:

1. To generate a report.
2. To create an extract file containing the items named in the EXTRACT command.
3. To generate a temporary subset of a data set. This temporary subset can then be referenced by other Inquiry commands.

This command has two forms. Option 1 creates a report or an extract file, and option 2 creates a temporary subset. These forms are discussed in the following paragraphs.

Option 1:

This form of the GENERATE command is used to create reports or extract files, or both.

If a report is generated, conditions for the data set used to generate the report must have been previously established by the SELECT, SET, or ATTACH commands. (Refer to the descriptions of these commands in this section.) If multiple copies of a report are desired, the GENERATE command may be repeated as many times as desired.

NOTE

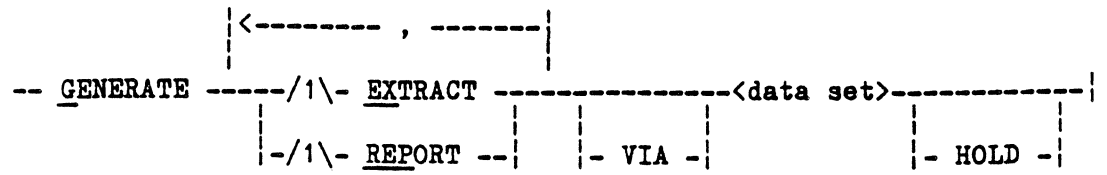
If other users are concurrently updating the data base, each report may be different.

The report generation process can be cancelled by entering the CANDE ?AX command.

If an extract file is created, file attributes for the extract file are set by the FILE command. When the extract is complete, the extract file is locked with CRUNCH. (That is, the file is closed and saved as a permanent disk file in such a way that the unused portion of the last row of disk space is returned to the system.) Any file with the current extract file title is removed.

Both an extract file and a report can be created using a single GENERATE command, if the same selection conditions are true for both.

Syntax:



Semantics:

EXTRACT

Generates an extract file.

REPORT

Generates a report.

VIA

May be included to improve readability.

<data set>

Contains the items to be reported or extracted.

HOLD

Holds the first output line of a report until a blank is entered. (This process allows a hardcopy terminal to be positioned at the top of a page so that the physical page control is synchronized with the report generation process.)

Examples:

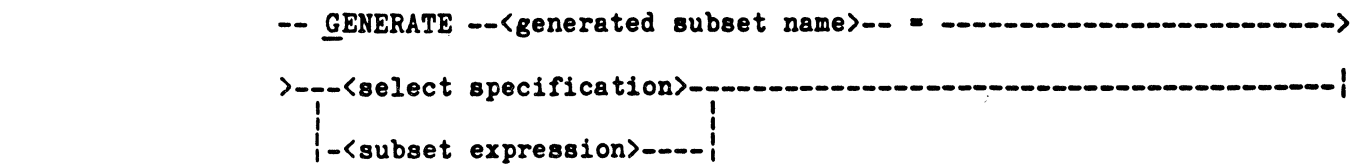
Examples illustrating the use of option-1 GENERATE commands may be found under "Inquiry REPORT" and "Inquiry EXTRACT" in Section 2, and under "REPORT Command" and "EXTRACT Command" in this section.

Option 2:

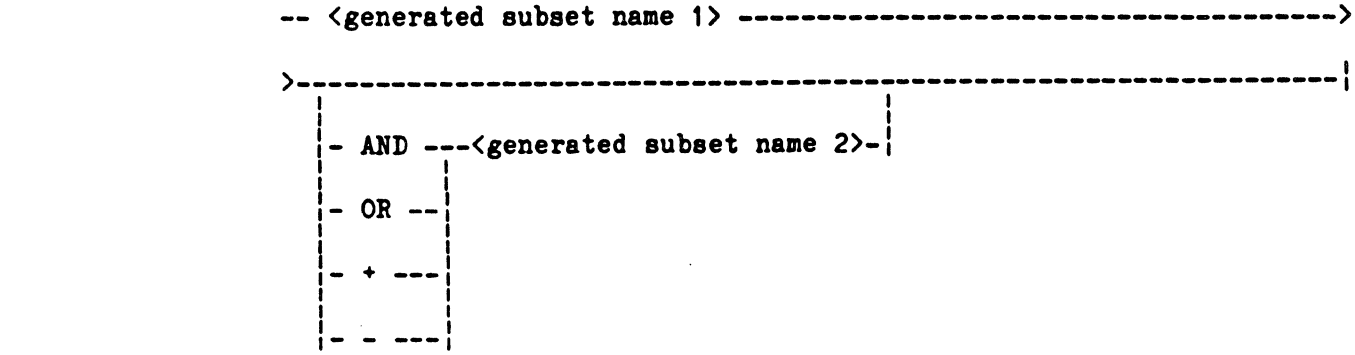
This form of the GENERATE command is used to create a temporary subset. The generated subset contains only the records that satisfy the condition at the time the subset is created. Although the text used to generate the generated subset can be saved, Inquiry discards the generated subset at the end of each session.

If UPDATE is allowed for the data base, and new records are created that satisfy the condition for a generated subset, the generated subset is updated to include the new records only after a RECALL is performed on the generated subset, followed by a REPEAT.

Syntax:



<subset expression>



Semantics:

<generated subset name>

Generates a temporary subset, which can then be referenced by SELECT and DISPLAY commands. Must be a unique identifier and must not be identical to any data base item name, define item name, virtual item name, generated subset name, or Inquiry command. May be used instead of <structure> in a SELECT or DISPLAY command.

Because temporary subsets are known only to INQUIRY, the ACCESSROUTINES cannot guarantee the integrity of the subset while the data base is undergoing change. Temporary subsets should be re-GENERATED following any update which affects the contents of the subset or the records within the subset. Because changes may be made by other user programs without detection by INQUIRY, generated subsets should never be used as the basis for updating a data base.

<select specification>

Defined under "SELECT Command" in this section.

<generated subset name 1>
<generated subset name 2>

Must refer to the same data set.

AND, OR,
+, -

May be used to form a generated subset from two existing generated subsets. These operations are defined in the following table, where G refers to the generated subset with <generated subset name>, G1 to the generated subset with <generated subset name 1>, and G2 to the generated subset with <generated subset name 2>.

G = G1 AND G2	Each entry in G is contained in both G1 and G2 (subset intersection).
G = G1 OR G2	Each entry in G is contained in either G1, or G2, or both (subset union).
G = G1 + G2	Each entry in G is contained in either G1 or G2, but not both (subset exclusive OR).
G = G1 - G2	Each entry in G is present in G1 and is absent from G2 (subset difference).

Examples:

The following GENERATE command creates the temporary subset RICHEMP from the data set EMP:

```
-> GENERATE RICHEMP = EMP-SALARY > 2000

   # WAIT
   #
```

```
-> DISPLAY RICHEMP

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     1  FIELDS      JON         533733557          720312
   2:TRUE             2250.00
```

128B

DMSII INQUIRY SOFTWARE OPERATION GUIDE

PCN 1164035-001 Release 3.6 December 1985

```
1:      2  HARDWICK     ALICE        647938292          701214
2:TRUE                3050.00
1:      5  CHAPMAN      GEORGE       166344267          710601
2:TRUE                2805.00
1:      6  MARLOWE      PHILLIP      867676865          551225
2:TRUE                2200.00
1:      9  JOYCE        JAIME        364277866          650714
2:TRUE                2450.00
#
```

The following GENERATE command creates the temporary subset OLDTIMERS from the data set EMP:

```
-> GENERATE OLDTIMERS = EMP-DATE-HIRED < 600101

   #

-> DISPLAY OLDTIMERS

   1:EMP-NO   EMP-LNAME    EMP-FNAME      EMP-SSN   EMP-DATE-HIRED
   2:EMP-SEC-CLEAR   EMP-SALARY
   ------------------------------------------------------------------
   1:      6  MARLOWE      PHILLIP      867676865          551225
   2:TRUE                2200.00
   1:      8  MITTY        WALTER       963443636          540303
   2:FALSE               2000.00
   #
```

The following GENERATE command creates the temporary subset TRUSTIES from the intersection of the subsets RICHEMP and OLDTIMERS:

```
-> GENERATE TRUSTIES = RICHEMP AND OLDTIMERS

   #

-> DISPLAY TRUSTIES

   1:EMP-NO   EMP-LNAME    EMP-FNAME      EMP-SSN   EMP-DATE-HIRED
   2:EMP-SEC-CLEAR   EMP-SALARY
   ------------------------------------------------------------------
   1:      6  MARLOWE      PHILLIP      867676865          551225
   2:TRUE                2200.00
   #
```

The following GENERATE command creates the temporary subset CRONIES from the union of the subsets RICHEMP and OLDTIMERS:

```
-> GENERATE CRONIES = RICHEMP OR OLDTIMERS

#

-> DISPLAY CRONIES

1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:      1  FIELDS       JON         533733557          720312
2:TRUE               2250.00
1:      2  HARDWICK     ALICE       647938292          701214
2:TRUE               3050.00
1:      5  CHAPMAN      GEORGE      166344267          710601
2:TRUE               2805.00
1:      6  MARLOWE      PHILLIP     867676865          551225
2:TRUE               2200.00
1:      8  MITTY        WALTER      963443636          540303
2:FALSE              2000.00
1:      9  JOYCE        JAIME       364277866          650714
2:TRUE               2450.00
#
```

The following GENERATE command creates the temporary subset PROTECTORS from the subset-exclusive-OR relationship between RICHEMP and OLDTIMERS:

```
-> GENERATE PROTECTORS = RICHEMP + OLDTIMERS

#

-> DISPLAY PROTECTORS

1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:      1  FIELDS       JON         533733557          720312
2:TRUE               2250.00
1:      2  HARDWICK     ALICE       647938292          701214
2:TRUE               3050.00
1:      5  CHAPMAN      GEORGE      166344267          710601
2:TRUE               2805.00
1:      8  MITTY        WALTER      963443636          540303
2:FALSE              2000.00
1:      9  JOYCE        JAIME       364277866          650714
2:TRUE               2450.00
#
```

The following GENERATE command creates the subset DEFENDERS from the subset difference between RICHEMP and OLDTIMERS:

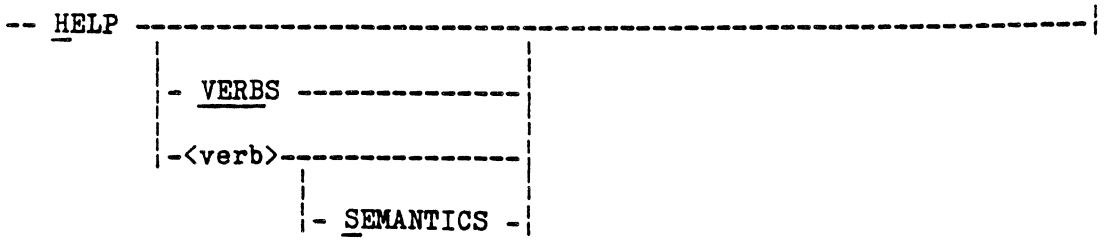
```
-> GENERATE DEFENDERS = RICHEMP - OLDTIMERS

#

-> DISPLAY DEFENDERS

1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:     1  FIELDS       JON          533733557          720312
2:TRUE               2250.00
1:     2  HARDWICK     ALICE        647938292          701214
2:TRUE               3050.00
1:     5  CHAPMAN      GEORGE       166344267          710601
2:TRUE               2805.00
1:     9  JOYCE        JAIME        364277866          650714
2:TRUE               2450.00
#
```

HELP Command

The HELP command displays either syntax or semantics for Inquiry statements or diagnostic information.

If the file DMINQ/HELP is not present, HELP information is not available and the system responds with a message indicating that this file is not present.

Syntax:



Semantics:

HELP

If no other options are specified, displays the most recent Inquiry error message.

VERB(S)

Displays a list of the Inquiry verbs.

<verb>

Displays the syntax of the <verb>. The entire name of the <verb> must be entered; abbreviations are not allowed.

SEMANTICS

Displays the semantics for the <verb>.

Examples:

The following HELP command displays syntax for the SORT command:

```
-> HELP SORT

   -- SORT ----------------------------------------!
      --    !                                      !
            ! !<-------------- , --------------! !
            ! !                                 ! !
            !----- CORE ----------- <integer> ---!
               !        ! !     !
               !- DISK -! !- = -!


   #
```

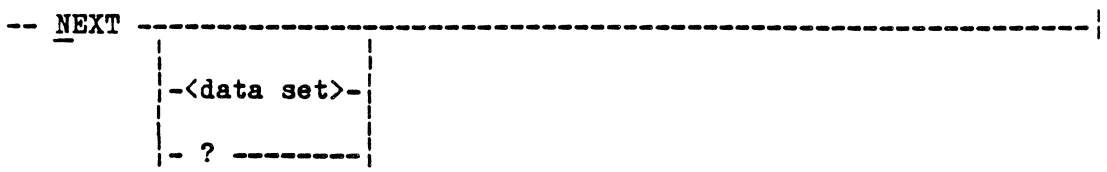
The following HELP command displays semantics for the SORT command:

```
-> HELP SORT SEMANTICS

   Display or alter resource parameters for the SORT intrinsic.
   #
```

NEXT Command

The NEXT command causes Inquiry to continue record selection and item display from the most recent stopping point.

Syntax:



Semantics:

NEXT

If no other options are specified, continues record selection and item display at the point that the most recently entered command stopped.

<data set>

Continues processing the most recently entered command at the point where the next record is to be selected for this data set. This point need not be the point at which the command stopped.

?

Displays the name of the next data set to be selected.

Example:

In the following example, the NEXT command is used to execute the DISPLAY command for the second record in ED-INST-NAME that satisfies the specified selection condition:

```
-> SELECT EMP THEN DISPLAY 1 EMP-LNAME,ED-INST-NAME AT ED-DATE > 550101

   EMP-LNAME    ED-INST-NAME
   ----------   ----------------------------
   FIELDS       UNIV OF MARYLAND
   #

-> NEXT

   EMP-LNAME    ED-INST-NAME
   ----------   ----------------------------
   FIELDS       CARLETON
   #
```

The following NEXT command executes the same DISPLAY command for the second record in EMP that satisfies the specified selection condition:

```
-> NEXT EMP

   EMP-LNAME    ED-INST-NAME
   ----------   ----------------------------
   HARDWICK     UCLA
   #
```

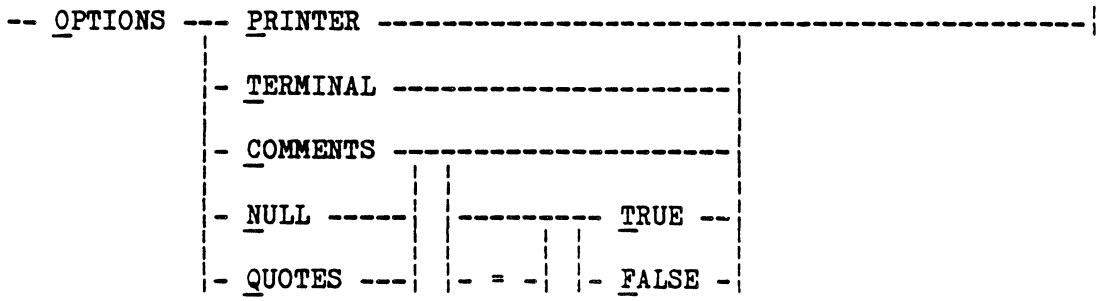
The following NEXT command displays the name of the next data set to be selected:

```
-> NEXT ?

   EMP-ED
   #
```

OPTIONS Command

The OPTIONS command controls the destination and contents of the output from DISPLAY and REPORT commands.

Syntax:



Semantics:

TERMINAL
PRINTER

Lists output generated by DISPLAY or REPORT commands on the user's terminal or the system line printer. TERMINAL is set by default. These options can be set at any time and remain set throughout the Inquiry session or until they are explicitly altered. If the output is directed to the printer, the number of lines per page and the number of characters per line may be specified using the PRINTER command. (Refer to PRINTER Command in this section.)

COMMENTS
COMMENTS TRUE

Displays DASDL comments along with the identifier when the SHOW command is used.

COMMENTS FALSE

Suppresses the display of DASDL comments. COMMENT is FALSE by default.

NULL
NULL TRUE

Displays all items regardless of value. Two hyphens are displayed for NULL items. NULL is TRUE by default.

NULL FALSE

Suppresses display of NULL items when TERMINAL FORMAT TAB or TERMINAL FORMAT SINGLE is used. (Refer to TERMINAL Command in this section.)

QUOTES
QUOTES TRUE

Requires alpha literals to be quoted alpha literals (that is, to begin and end with quotation marks). QUOTES is TRUE by default.

QUOTES FALSE

Allows quotation marks to be omitted from alpha literals.

Care must be taken in using alpha literals when QUOTES is FALSE. Alpha literals in UPDATE and CREATE commands must always be quoted alpha literals. The TITLE command accepts either quoted alpha literals or unquoted alpha literals. However, the TITLE options DATE and TIME are not interpreted as unquoted alpha literals, and alpha literals containing special characters or embedded blanks must be quoted. (Also refer to the discussion of unquoted alpha literals under <selection condition> in "Common Syntactic Items.")

Quoted alpha literals may be used at any time, even if QUOTES is FALSE.

=

May be included to improve readability.

Examples:

The following example illustrates terminal output when the OPTIONS PRINTER command is executed:

```
-> OPTIONS PRINTER

   #
```

```
-> DISPLAY 2 EMP

   # DISPLAY # 1 ON PRINTER
   # 2 RECORDS FOR EMP PRINTED.
   #
```

In the following example, an OPTIONS COMMENTS is executed to display the DASDL comment, in quotes, along with the DASDL description of the embedded data set EMP-ED:

```
-> SHOW EMP-ED

   EMP-ED OF EMP DATA SET
   ITEMS:
      ED-DATE NUMBER (6)
      ED-INST-NAME ALPHA (30)
      ED-INST-CODE NUMBER (4)
      ED-COURSE ALPHA (10)
      ED-GPA NUMBER (3,2)
   SETS:
      ED-DATE-SET KEY ED-DATE
   #

-> OPTIONS COMMENTS

   #

-> SHOW EMP-ED

   EMP-ED OF EMP DATA SET "EDUCATIONAL RECORDS FOR EACH EMPLOYEE"
   ITEMS:
       ED-DATE "YYMMDD" NUMBER (6)
       ED-INST-NAME ALPHA (30)
       ED-INST-CODE NUMBER (4)
       ED-COURSE ALPHA (10)
       ED-GPA NUMBER (3,2)
   SETS:
     ED-DATE-SET KEY ED-DATE
   #
```

In the following example, an OPTION QUOTES = FALSE command is executed so that the alpha literal "BURNS" is not required to appear in quotes:

```
-> OPTIONS QUOTES = FALSE

   #

-> DISPLAY EMP AT EMP-LNAME = BURNS
```

```
1:EMP-NO  EMP-LNAME . EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
2:EMP-SEC-CLEAR  EMP-SALARY
-------------------------------------------------------------------
1:     7  BURNS       GEORGINA    323313145          741010
2:FALSE             1750.00
#
```

PRINTER Command

The PRINTER command displays or alters the printer file attributes assigned by Inquiry. These attributes are active only if OPTIONS PRINTER is set. (Refer to OPTIONS Command in this section.)

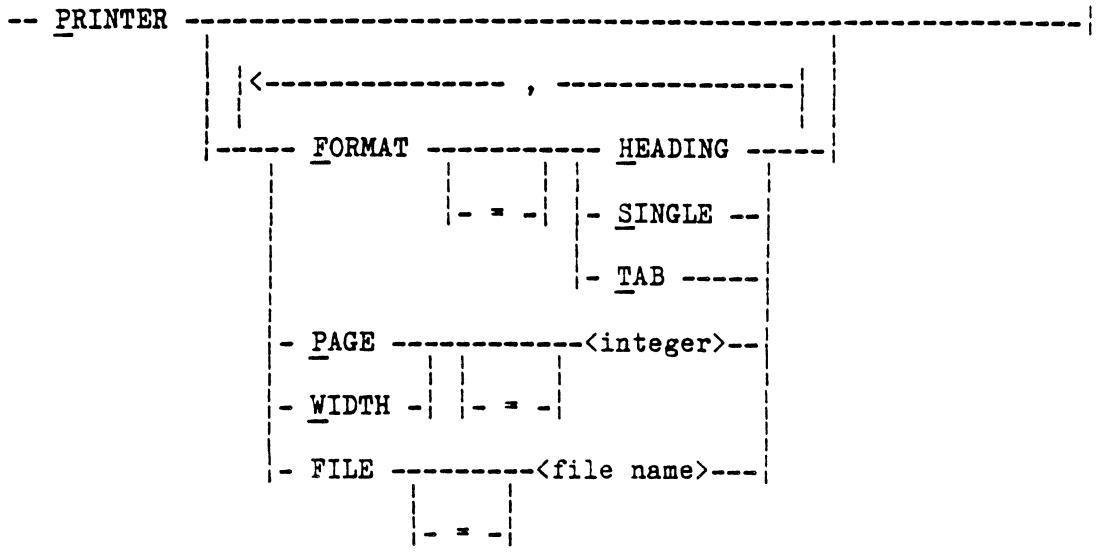
The PRINTER command may specify a different INTNAME attribute for the printer file. If so, any file attributes associated with the specified file at the time Inquiry is initiated are associated with the printer file. If the printer file is open, it is closed, and its title and INTNAME are both set to the specified file name. If no file-equation WFL statement is supplied for this file name, the file reverts to the default printer file with the specified file name as its title.

Any special requirements for the printer file should be anticipated. FILE is not required to produce normal printer files, but can be used when additional printer file characteristics are sought.

NOTE

PRINTER is the original INTNAME of the printer file. The printer file is not actually opened until the file is explicitly required by an Inquiry command; consequently, this process may be repeated as often as necessary.

Syntax:



Semantics:

PRINTER

If no other options are specified, displays the current printer attributes.

FORMAT

Selects one of the three specified output formats.

HEADING
SINGLE
TAB

Refer to "Formatting Attributes" in Section 2.

PAGE

Specifies the number of lines displayed on each page.

WIDTH

Specifies the number of characters displayed on each line.

FILE

Changes the INTNAME of the printer file to <file name>.

=

May be included to improve readability.

Examples

The following PRINTER command sets the page width to 55 columns and the number of lines printed to 10:

```
-> PRINTER PAGE = 10, WIDTH = 55

   #
```

The following examples show Inquiry display messages from the generation of printer output and the printer output using the specifications from the previous example in each of the three output formats:

```
-> PRINTER FORMAT = TAB

   #

-> DISPLAY 1 EMP


   # DISPLAY # 1 ON PRINTER
   # 1 RECORDS FOR EMP PRINTED.
   #


    ***DISPLAY  # 1
      EMP-NO=1  EMP-LNAME=FIELDS
      EMP-FNAME=JOHN          EMP-SSN=533733557
      EMP-DATE-HIPED=720312   EMP-SEC-CLEAR=TRUE
      EMP-SALARY=5000.00

-> PRINTER FORMAT = HEADING

   #
```

```
-> DISPLAY 1 EMP

   # DISPLAY # 2 ON PRINTER
   # 1 RECORDS FOR EMP PRINTED.
   #


    *** DISPLAY # 2
    1:EMP-NO   EMP-LNAME   EMP-FNAME     EMP-SSN
    2:EMP-DATE-HIRED EMP-SEC-CLEAR EMP-SALARY
    ---------------------------------------------
    1:     1 FIELDS       JON          533733557
    2:       720312  TRUE                5000.00


-> PRINTER FORMAT = SINGLE

   #


-> DISPLAY 1 EMP

   # DISPLAY # 3 ON PRINTER
   # 1 RECORDS FOR EMP PRINTED.
   #


    *** DISPLAY # 3
      EMP-NO = 1
      EMP-LNAME = FIELDS
      EMP-FNAME=  JON
      EMP-SSN =533733557
      EMP-DATE-HIRED = 720312
      EMP-SEC-CLEAR = TRUE
      EMP-SALARY = 5000.00
```

QUIT Command

The QUIT command terminates an Inquiry session.

Before the session is terminated, the text of all define, generate, and virtual items must be saved or discarded. If QUIT is entered and define, generate or virtual items have been added, deleted or modified but not saved, or if the declarations are saved in any file other than the default file, the following message is displayed:

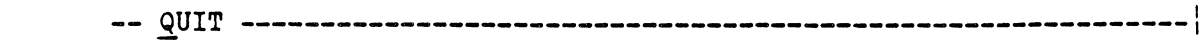
```
#DECLARATIONS NOT SAVED IN DEFAULT FILE
```

If a SAVE command is then entered, the declarations are saved and the Inquiry session is terminated. If QUIT is entered again, the Inquiry session is terminated and the declarations are discarded.

After QUIT is entered, Inquiry responds with the following message before closing the data base:

```
#TERMINATING Inquiry
```

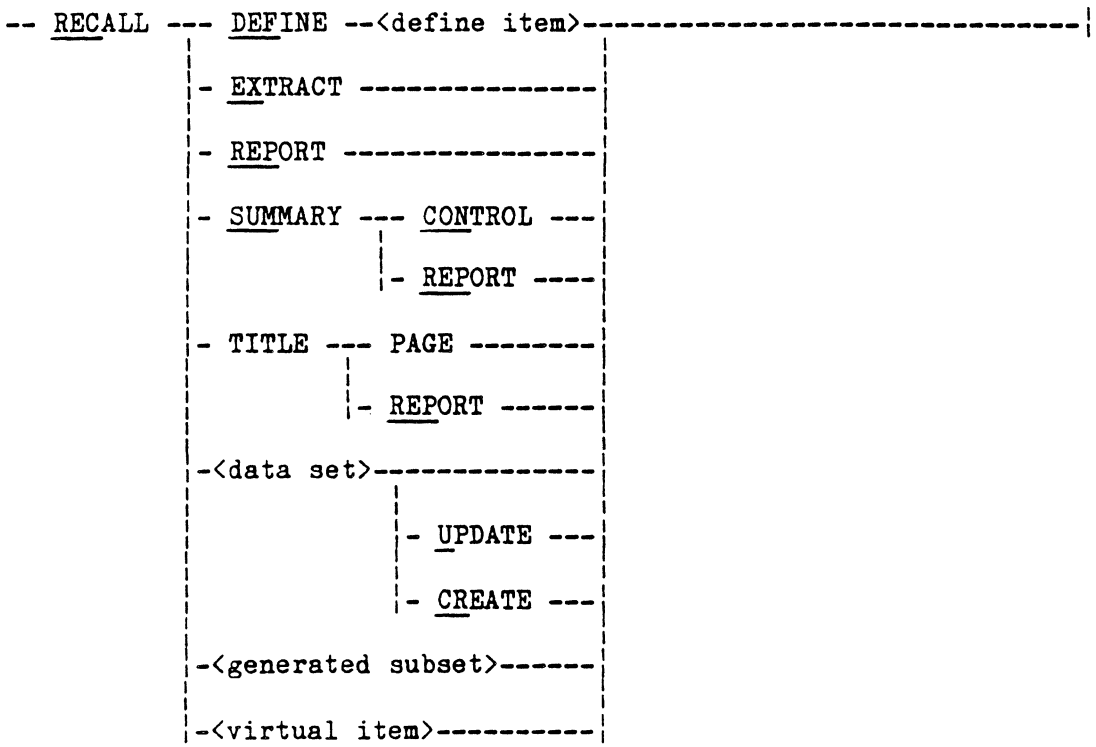
Syntax:



RECALL Command

The RECALL command loads the current buffer with the text of the specified command, generated subset, or item and displays this text on the terminal. Once the text has been recalled, it may be modified using the EDIT command or executed using the REPEAT command.

Syntax:



Semantics:

DEFINE <define item>

Loads and displays text associated with the <define item>. Each command of a multiple-command define is displayed on a separate line, preceded by a line number and a colon (:).

EXTRACT

Loads and displays previously entered EXTRACT commands.

REPORT

Loads and displays previously entered REPORT commands. (In some instances, changes to the report list may require changes to the SUMMARY command.)

SUMMARY CONTROL
SUMMARY REPORT

Loads and displays text associated with the specified SUMMARY command.

TITLE PAGE
TITLE REPORT

Loads and displays text associated with the specified TITLE command.

<data set>

Loads and displays the text of SELECT and DISPLAY commands associated with this data set.

<data set> UPDATE
<data set> CREATE

Loads and displays text of UPDATE or CREATE commands associated with <data set>.

<generated subset>

Loads and displays the text of the associated GENERATE command.

<virtual item>

Loads and displays the text of the associated VIRTUAL command.

Examples:

The following RECALL command reloads and displays the text of the DEFINE command that created the define HIRERANGE (in the second example under DEFINE Command in this section):

```
-> RECALL DEFINE HIRERANGE
```

```
 DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE
AND EMP-DATE-HIRED < LDATE
#
```

The following RECALL command reloads and displays the text of the GENERATE command that created the generated subset RICHEMP (in the first example under GENERATE Command in this section):

```
-> RECALL RICHEMP

   GENERATE RICHEMP = EMP-SALARY > 2500
   #
```

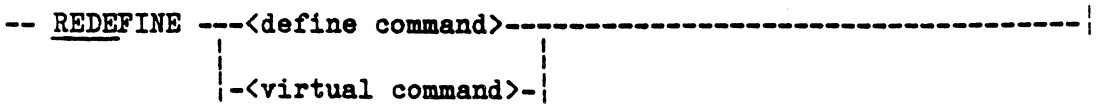
The following RECALL command reloads and displays the text of the VIRTUAL command that created the virtual item EMP-YEAR-SALARY (in the example under "VIRTUAL Command" in this section):

```
-> RECALL EMP-YEAR-SALARY

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12
   #
```

REDEFINE Command

The REDEFINE Command allows the changing of a define or a virtual while the define or virtual is in use.

Syntax:



Semantics:

<define command>

Changes the text associated with a define identifier. Inquiry will forget the old text and remember the new text. Any command using the define identifier will now reference the new text.

<virtual identifier>

Changes the vale of a virtual identifier. Inquiry will forget the old value and calculate the new value. This command is similiar to clearing the virtual and re-entering it with a new value except that references to this virtual by other virtuals are not checked.

Examples:

The following REDEFINE command changes the text associated with the define SHOWDEX:

```
-> DEFINE SHOWDEX=SHOW DECLARATIONS
   #
-> REDEFINE DEFINE SHOWDEX=SHOW DEFINES;SHOW VIRTUALS;
   #
   01 DEFINE SHOWDEX = SHOW DEFINES;
   02 SHOW VIRTUALS;
   #
```

The following REDEFINE command shows the difference between the use of the REDEFINE command and the use of the combination of the CLEAR and VIRTUAL commands.

```
-> VIRTUAL RAISE-PCT = .10
   #
-> VIRTUAL SALARY-W-RAISE = EMP-SALARY * (1 + RAISE-PCT)
   #
-> CLEAR RAISE-PCT
   #VIRTUAL RAISE-PCT IS REFERENCED BY VIRTUAL SALARY-W-RAISE
   #VIRTUAL REFERENCED BY ANOTHER VIRTUAL CANNOT BE CLEARED
   #
-> SHOW RAISE-PCT

   VIRTUAL RAISE-PCT = .10
   #
-> REDEFINE VIRTUAL RAISE-PCT = .15
   #
-> SHOW VIRTUALS

   VIRTUAL RAISE-PCT = .15
   VIRTUAL SALARY-W-RAISE = EMP-SALARY * (1 + RAISE-PCT)
   #
```

REPEAT Command

The REPEAT command causes the command in the current buffer to be executed.

If a DEFINE, VIRTUAL, or GENERATE command is presently in the current buffer, REPEAT causes Inquiry to remember the text associated with that command. If RECALL is entered to place the text of a define item, virtual item, or generated subset in the current buffer, and this text has been altered, REPEAT causes Inquiry to discard the old text and retain the altered text. In this situation, if REPEAT is not entered, the unaltered text is retained.

REPEAT may be used to modify and execute the command in the current buffer within one operation; this operation is identical to an EDIT followed by a REPEAT. (Refer to the examples below.) If more than one line of text is present, specification of a line number before the first delimiter causes the editing specifications to refer to that particular line. If this line number does not correspond to one of the lines displayed, a syntax error occurs. The line numbers are not saved. Each number indicates only the relative position of a line within the entire text; therefore, if subsequent changes occur, such as command insertions or deletions, the line numbers change. For this reason, whenever subsequent editing operations take place, new text and line numbers are automatically displayed.

Similarly, Inquiry remembers the text of the most recent SELECT or DISPLAY command associated with each data set. Thus, if a REPEAT is entered for a specified data set, statements associated with the data set or with attached embedded data sets are executed.

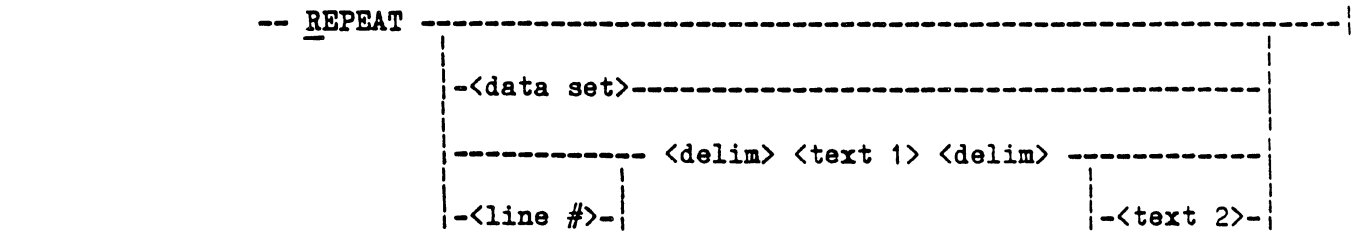
The SET command may be used to associate a REPEAT command with a particular data set. For example, the following SET statement would associate a REPEAT <data set 2> command with <data set 1>:

```
SET <data set 1> TO REPEAT <data set 2>
```

In this case, if a REPEAT <data set 1> statement is subsequently executed, the commands associated with <data set 2> are automatically executed at the same time the REPEAT <data set 1> command is executed. In Inquiry terms, automatic looping has been established between <data set 1> and <data set 2>. <data set 1> and <data set 2> in this example need not be related. In fact, the establishment of looping using the REPEAT command is the only way in which Inquiry allows access to

information contained in records of different disjoint data sets. (Refer to "RELATED RECORDS ACROSS DATA SETS" in Section 2.)

Syntax:



Semantics:

REPEAT

If no other options are specified, executes the command in the current buffer.

<data set>

Executes the most recent SELECT or DISPLAY command associated with this data set.

<delim>

Any special character not contained in <text 1>. Both <delim>s must be the same character.

<text 1>
<text 2>

If <text 2> is specified, replaces <text 1> by <text 2>; otherwise, eliminates <text 1>. Only the first occurrence of <text 1> is affected.

Examples:

In the following example, the REPEAT command is used to edit the define item SHOWZIP and replace the original text with the new text:

```
-> RECALL DEFINE SHOWZIP
```

```
DEFINE SHOWZIP = D ADR-ZIP AT ADR-EMP-NO < 6
#

-> REPEAT /ADR-EMP-NO < 6/ADR-EMP-NO < 7

DEFINE SHOWZIP = D ADR-ZIP AT ADR-EMP-NO < 7
#
```

In the following example, a SET command is used to establish a REPEAT command for the JOB data set and, thus, to establish automatic looping between the JOB data set and the EMP data set:

```
-> SET JOB TO DISPLAY JOB AT JOB-TYPE=2;SET JOB TO SORT JOB-TYPE

#

-> SET JOB TO REPEAT EMP;SET EMP TO DISPLAY 3 EMP-NAME,EMP-DATE-HIRED

#

->  REPEAT JOB

#WAIT
1:JOB-COUNT  JOB-TYPE  JOB-CODE  JOB-TITLE
2:TYPE-SPEED  SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD
3:DRIVING-REQD  JOB-COMMENT
-------------------------------------------------------------------
1:        1         2      3333  SALESMAN
2:        --                 --  --           TRUE
3:FALSE          MINUMUM 2 YRS FIELD EXP
  EMP-LNAME   EMP-FNAME   EMP-DATE-HIRED
  ----------  ----------  --------------
  FIELDS      JON                720312
  HARDWICK    ALICE              701214
  JONSON      BEN                620115
#
```

For additional examples, refer to "RELATED RECORDS ACROSS DATA SETS" in Section 2 and to "ATTACH Command" and "EDIT Command" in this section.

REPORT Command

The REPORT command specifies the items to be listed on an Inquiry report and controls the format of the report.

The following items may appear in a report:

1. Any data base item except a group name. If the item has an OCCURS clause, the subscript may be an arithmetic expression.
2. A virtual item.
3. An arithmetic function.
4. An arithmetic expression.

Items specified in a REPORT command are printed in columnar format. If these items cannot fit on the current output device line, Inquiry displays a warning message. The output device and/or device attributes must be changed before the report is generated or an error message will be given.

If the specified items for a report directly or indirectly (for example, through expressions or virtuals) reference a data set other than the data set specified in the GENERATE REPORT <data set> command, and no current record is present for that data set, then the specified items are undefined on the report and appear as one or two hyphens (-).

Certain items specified in the REPORT command may be designated report control items. These items appear on the report in the following form:

<item>:<value>

A control break occurs when the value of any report control item changes between two adjacent records. At this time, processing of records stops and any summary information requested by a previous SUMMARY CONTROL command is tabulated. (Refer to SUMMARY Command in this section.) To provide proper report generation control, the SORT option should be used to sort the selected records in order of the report control items, if the records have not been selected in that order. The SET command should be used to specify the sort criteria, in order to avoid redundant calls on the system SORT intrinsic.

The starting column and column width may be specified for each item in the report. If no starting column is specified for an item, the column starts one character beyond the point where the previous column left off. A COLUMN OVERLAP message is displayed if the specified starting column is less than one position beyond the ending of the previous column.

If no column width is specified, the column width is assumed to be the largest of either the number of characters in the column name or the number of characters required to present the data value without loss of significance. In addition, the TRUNCATE option is ignored.

If the specified column width is not large enough to hold the column name, the specification is ignored and a column width large enough to hold the column name is assumed. If the specified column width is not large enough to hold the data to be formatted, the default action depends on the data type, as follows:

Numeric

> The value is rounded to fit within the specified column width. If the value cannot be rounded without loss of leading significant digits, the column is filled with asterisks (*).

Boolean

> The left-most portion of the TRUE or FALSE is placed in the column.

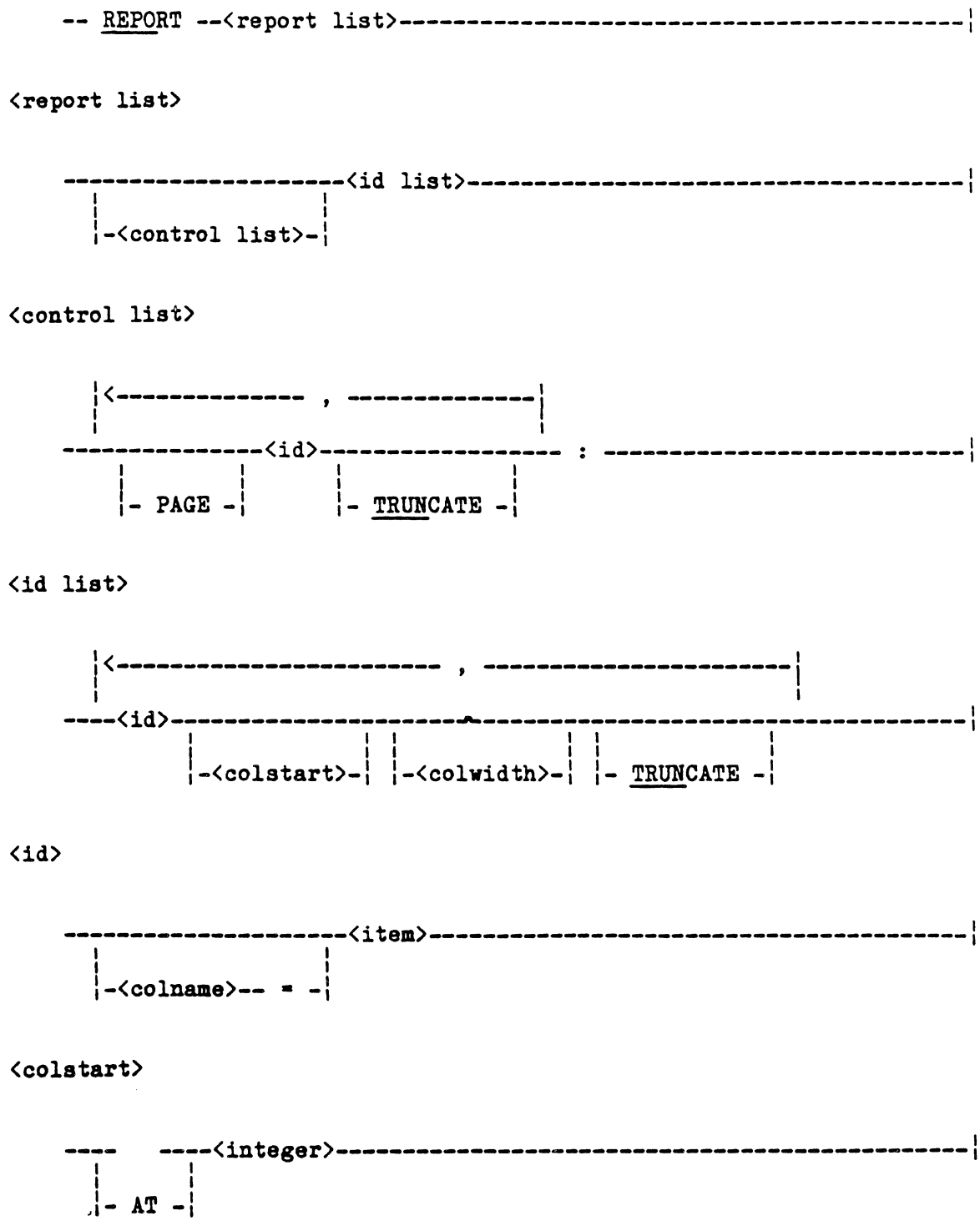
Alpha

> If the TRUNCATE option is not specified, the entire alpha variable is printed, even if multiple report lines are required to do so. If the TRUNCATE option is specified, any characters in the alpha variable that do not fit are truncated from the right.

The column width may be specified much larger than necessary so that items are spaced evenly across the report page.

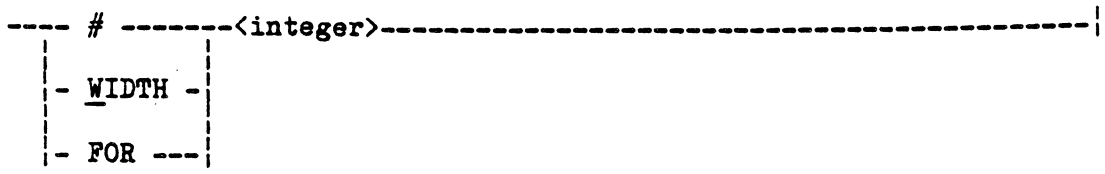
Note that the column number added to the column width must not exceed the device line size. Otherwise, the error message "REPORT TOO LARGE FOR PAGE WIDTH" is generated, even if the actual data items can fit.

Syntax:


-- REPORT --<report list>
<report list>
<id list>
-<control list>-
<control list>
,
<id>
:
- PAGE -
- TRUNCATE -
<id list>
,
<id>
-<colstart>-
-<colwidth>-
- TRUNCATE -
<id>
<item>
-<colname>-- = -
<colstart>
- AT -
<integer>

Note that <colstart> added to <colwidth> must not exceed the device line size.

<colwidth>



Note that <colwidth> added to <colstart> must not exceed the device line size.

Semantics:

<report list>

Lists items to be reported.

<control list>

Specifies report control items. Report control items group records with the same value for that item on the report.

<id list>

Specifies items to appear in columns.

<id>

The <item> name or specified <colname>.

:

Separates report control items from other displayed items.

PAGE

Causes a new page to be started if a control break occurs.

TRUNCATE

For alpha variables, specifies that any characters that do not fit on a line are truncated from the right. If TRUNCATE is not specified, the entire alpha variable is printed, using multiple lines if necessary; however, trailing blanks are suppressed. (This option is ignored for non-alpha variables.)

<colstart>

Specifies the starting position of a column on the report page.

<colwidth>

Specifies the number of character positions in which a value is formatted.

<colname>

An alphanumeric literal used to specify the name printed as the column name on the report. <colname> must begin with a letter and cannot be longer than 17 characters. If <colname> is not specified, the <item> name is used.

Examples:

The following SET command establishes a SELECT command containing a selection expression to be used when a report is generated. The SELECT command also sorts the selected records in order of JOB-CODE:

```
-> SET JOB TO SELECT JOB-MNG-SUBSET, SORT JOB-CODE

   #
```

The REPORT command is then used to specify the report format. The item JOB-TITLE is specified as a report control item. In addition, the column headings in the report for the items JOB-CODE and JOB-COMMENT appear as CODE and COMMENT, respectively. The job codes in the report appear beginning in column 5 and extend for 4 columns. The job comments in the report appear beginning in column 15 and extend for 50 columns. The following REPORT command contains these specifications:

```
-> REPORT JOB-TITLE:CODE=JOB-CODE AT 5 FOR 4,COMMENT=JOB-COMMENT %
   #%
-> AT 15 FOR 50
```

The report is then generated using a GENERATE command:

```
-> GENERATE REPORT VIA JOB

   # STARTING REPORT

                                                              PAGE 1
   JOB-TITLE : PROJECT MANAGER
       CODE      COMMENT
       ----      ------------------------------------------------
       5000      3 YRS MANAGEMENT EXP PLUS DEGREE

   JOB-TITLE : ADM ASSIST
       CODE      COMMENT
       ----      ------------------------------------------------
       7000      --

   JOB-TITLE : SALES MANAGER
       CODE      COMMENT
       ----      ------------------------------------------------
       7777      5 YRS SALES EXP PLUS DEGREE


   JOB-TITLE : CONTROLLER
       CODE      COMMENT
       ----      ------------------------------------------------
       8888      --

             .
             .
             .
                                                              PAGE 2
  JOB-TITLE : VICE PRESIDENT
       CODE      COMMENT
       ----      ------------------------------------------------
       9999      --

   # REPORT COMPLETED
```

The REPORT command in the following example contains the same specifications as that in the previous example, except that a PAGE option is added to start a new page each time a control break occurs:

```
-> REPORT PAGE JOB-TITLE:CODE=JOB-CODE AT 5 FOR 4,COMMENT=%
   #%
-> JOB-COMMENT AT 15 FOR 50

   #

-> GENERATE REPORT VIA JOB
```

```
# STARTING REPORT
                                                              PAGE 1
JOB-TITLE : PROJECT MANAGER
    CODE      COMMENT
    ----      -------------------------------------------------
    5000      3 YRS MANAGEMENT EXP PLUS DEGREE

              .
              .
              .
                                                              PAGE 2
JOB-TITLE : ADM ASSIST
    CODE      COMMENT
    ----      -------------------------------------------------
    7000      --

              .
              .
              .
                                                              PAGE 3
JOB-TITLE : SALES MANAGER
    CODE      COMMENT
    ----      -------------------------------------------------
    7777      5 YRS SALES EXP PLUS DEGREE

              .
              .
              .

                                                              PAGE 4
JOB-TITLE : CONTROLLER
    CODE      COMMENT
    ----      -------------------------------------------------
    8888      --

              .
              .
              .
                                                              PAGE 5
JOB-TITLE : VICE PRESIDENT
    CODE      COMMENT
    ----      -------------------------------------------------
    9999      --

# REPORT COMPLETED
```

In the following example, the column specifications for the report generated in the first example are edited so that the values appearing in the report must be broken over a number of lines:

```
-> RECALL REPORT

   REPORT JOB-TITLE:CODE=JOB-CODE AT 5 FOR 4,COMMENT=JOB-COMMENT AT
    15 FOR 50

   #

-> EDIT /FOR 50/FOR 20

   REPORT JOB-TITLE:CODE=JOB-CODE AT 5 FOR 4,COMMENT=JOB-COMMENT AT
    15 FOR 20

   #

-> REPEAT

   #

-> GENERATE REPORT VIA JOB

   # STARTING REPORT
                                                              PAGE 1
   JOB-TITLE : PROJECT MANAGER
       CODE      COMMENT
       ----      --------------------
       5000      3 YRS MANAGEMENT EXP
                 PLUS DEGREE


   JOB-TITLE : ADM ASSIST
       CODE      COMMENT
       ----      --------------------
       7000      --


   JOB-TITLE : SALES MANAGER
       CODE      COMMENT
       ----      --------------------
       7777      5 YRS SALES EXP PLUS
                 DEGREE
              .
              .
              .
                                                              PAGE 2
   JOB-TITLE : CONTROLLER
       CODE      COMMENT
       ----      --------------------
       8888      --

   JOB-TITLE : VICE PRESIDENT
       CODE      COMMENT
       ----      --------------------
       9999      --
```

```
# REPORT COMPLETED
#
```

If the TRUNCATE option had been specified for the COMMENT column in the preceding example, the report would be output as follows:

```
# STARTING REPORT
                                                          PAGE 1

JOB-TITLE : PROJECT MANAGER
    CODE      COMMENT
    ----      -------------------
    5000      3 YRS MANAGEMENT EXP

JOB-TITLE : ADM ASSIST
    CODE      COMMENT
    ----      -------------------
    7000      --

JOB-TITLE : SALES MANAGER
    CODE      COMMENT
    ----      -------------------
    7777      5 YRS SALES EXP PLUS

JOB-TITLE : CONTROLLER
    CODE      COMMENT
    ----      -------------------
    8888      --

          .
          .
          .
                                                          PAGE 2
JOB-TITLE : VICE PRESIDENT
    CODE      COMMENT
    ----      -------------------
    9999      --

# REPORT COMPLETED
#
```

In the following example, the report generated from EMP includes the result of the arithmetic function SQRT over the virtual item EMP-YEAR-SALARY for each employee in EMP:

```
-> SELECT EMP

#
```

```
-> REPORT EMP-LNAME,EMP-SALARY,SQRT(EMP-YEAR-SALARY)

   #
-> GENERATE REPORT VIA EMP

   # STARTING REPORT
                                                                PAGE 1
    EMP-LNAME  EMP-SALARY              SQRT
    ---------- ---------- ------------------
    FIELDS        2000.00      154.919333848
    KRANTZ        3050.00      191.311264697
    JONSON        1771.56      145.803703657
    DEKKER        1950.00      152.970585408
    CHAPMAN       2805.00      183.466618218
    MARLOWE       2200.00      162.480768093
    BURNS         1750.00      144.913767462
    MITTY         2000.00      154.919333848
    JOYCE         2450.00      171.464281995
    STACK         4500.00      232.379000773

   # REPORT COMPLETED
   #
```

"SUMMARY Command" and "TITLE Command" in this section contain additional examples using the reports generated in the preceding examples. (Refer also to Inquiry REPORT in Section 2.)

RESTORE Command

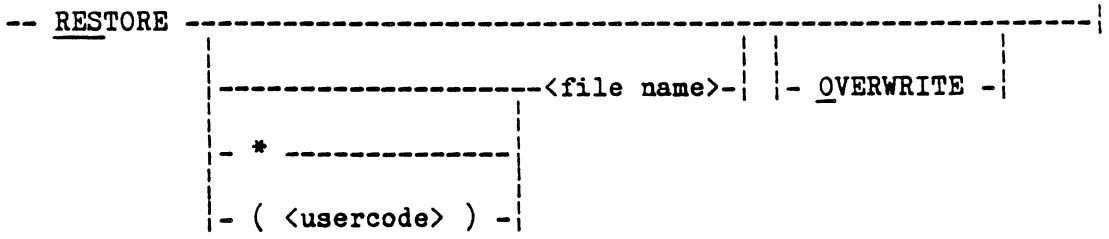
The RESTORE command reloads previously saved declarations. If a file is specified in a RESTORE command, the contents of that file are restored. The specified file must have been created previously using a SAVE command. If no file name is specified, Inquiry restores declarations from the default definitions file. (The title of this file is specified when running BUILDINQ; refer to "Inquiry Generation" in Appendix C.) If the declarations file was saved under a usercode other than that of the Inquiry user, or under the system file directory (using an "*"), the usercode or asterisk must be specified in the RESTORE command. Nonprivileged users cannot access files saved in the system file directory (*) or under a usercode other than their own.

Normally, when a declaration created during an Inquiry session and a saved declaration have the same name, RESTORE retains the existing declaration and discards the saved declaration. However, the RESTORE command may specify that the saved declaration is to replace the existing declaration.

Following a RESTORE, the SHOW DEFINES, SHOW GENERATES, and SHOW VIRTUALS statements can be used to display the restored declarations.

RESTORE does not execute any of the declarations or regenerate any subsets created by the GENERATE command. The RECALL and REPEAT commands can be used to re-establish generated subsets.

Syntax:



Semantics:

RESTORE

If no other options are specified, reloads declarations from the default definitions file.

<file name>

Reloads saved declarations in this file.

OVERWRITE

Replaces existing declaration with the original saved declaration.

*

Specified file containing declarations to be reloaded is on the system directory.

(<usercode>)

Specified file containing declarations to be reloaded is on this usercode.

Examples:

The following RESTORE command reloads the virtual item, generated subset, and define item saved in the default definitions file:

```
RESTORE
```

The following RESTORE command would reload the definitions stored in the file DEFAULTDEF on usercode MADDOG:

```
RESTORE (MADDOG) DEFAULTDEF
```

The following RESTORE command would reload the definitions stored in the file DEFDEFAULT on the system directory:

```
RESTORE *DEFDEFAULT
```

The following example illustrates the use of the RESTORE OVERWRITE command. First, the saved virtual item EMP-YEAR-SALARY is recalled, edited, and repeated (to preserve the edited version of the virtual item). The RESTORE OVERWRITE command is then used to replace the existing (that is, newly changed) virtual item EMP-YEAR-SALARY with the original virtual item saved in the default definitions file.

```
-> RECALL EMP-YEAR-SALARY

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12 + 2500
   #

-> EDIT /2500/3000

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12 + 3000
   #

-> REPEAT

   #

-> SHOW VIRTUAL

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12 + 3000
   #

-> RESTORE OVERWRITE

   #

-> SHOW VIRTUAL

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12 + 2500
   #
```

SAVE Command

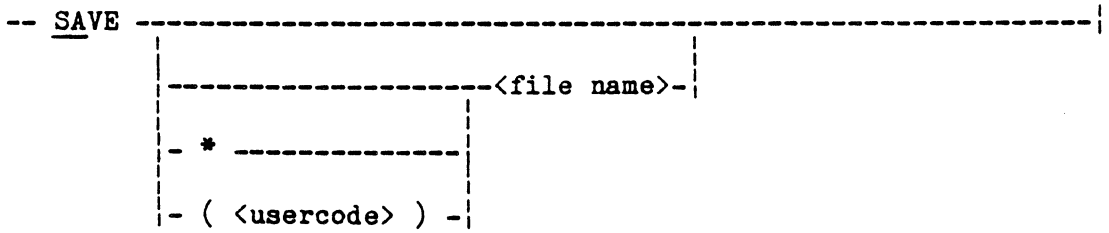
The SAVE command stores the text of DEFINE, VIRTUAL, and GENERATE commands. Use of this command allows a user to declare these statements during one Inquiry session and save them for use during subsequent sessions.

Inquiry saves the most current declarations. If a SAVE is entered after declarations have been added, deleted, or modified, a new save file is created and the old file, if any, is removed.

If the SAVE command specifies a file name, Inquiry saves the declarations in a file with the specified name. If no file name is specified, the declarations are saved in the definitions file. (The title of the definitions file is specified when running BUILDINQ; refer to "Inquiry Generation" in Appendix C of this manual.)

If an asterisk (*) precedes the specified file name, the file is saved in the system file directory. If a usercode precedes the specified file name, the file is saved under that usercode. (In these cases, the file is available only to privileged users because only privileged users are allowed to create or save system files or files under other usercodes.) If neither an asterisk nor a usercode is specified, the definitions file is saved under the terminal user's usercode.

Syntax:



Semantics:

SAVE

If no other options are specified, stores text of DEFINE, VIRTUAL, and GENERATE commands.

<file name>

Declarations are saved in a file with this name.

*

Declarations are saved in the specified system file.

(<usercode>)

Declarations are saved in the specified file under this usercode.

Examples:

In the following example, the SAVE command is used to save the newly created define item SHOWZIP. Then, a message is displayed indicating that the old definitions file that did not contain SHOWZIP is removed. It is replaced by the new definitions file containing SHOWZIP:

```
-> SHOW DEFINES

   DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE AND
    EMP-DATE-HIRED < LDATE
   #

-> DEFINE SHOWZIP = D ADR-ZIP

   #

-> SAVE

   #6707 (SOFTDOC)DEFINITIONS/DIV68 REMOVED ON SOFTDOC PK63.
   #

-> SHOW DEFINES

   DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE AND
    EMP-DATE-HIRED < LDATE

   DEFINE SHOWZIP = D ADR-ZIP
   #
```

In the following example, the SAVE command is entered at the time the Inquiry session is terminated to save the declarations created during that session:

```
-> QUIT

   #DECLARATIONS NOT SAVED IN DEFAULT FILE

-> SAVE

   #6717 (SOFTDOC)DEFINITIONS/DIV68 REMOVED ON SOFTDOC PK63.
   #TERMINATING Inquiry
   #ET=2:20.6 PT=13.5 IO=14.0
```

SET Command

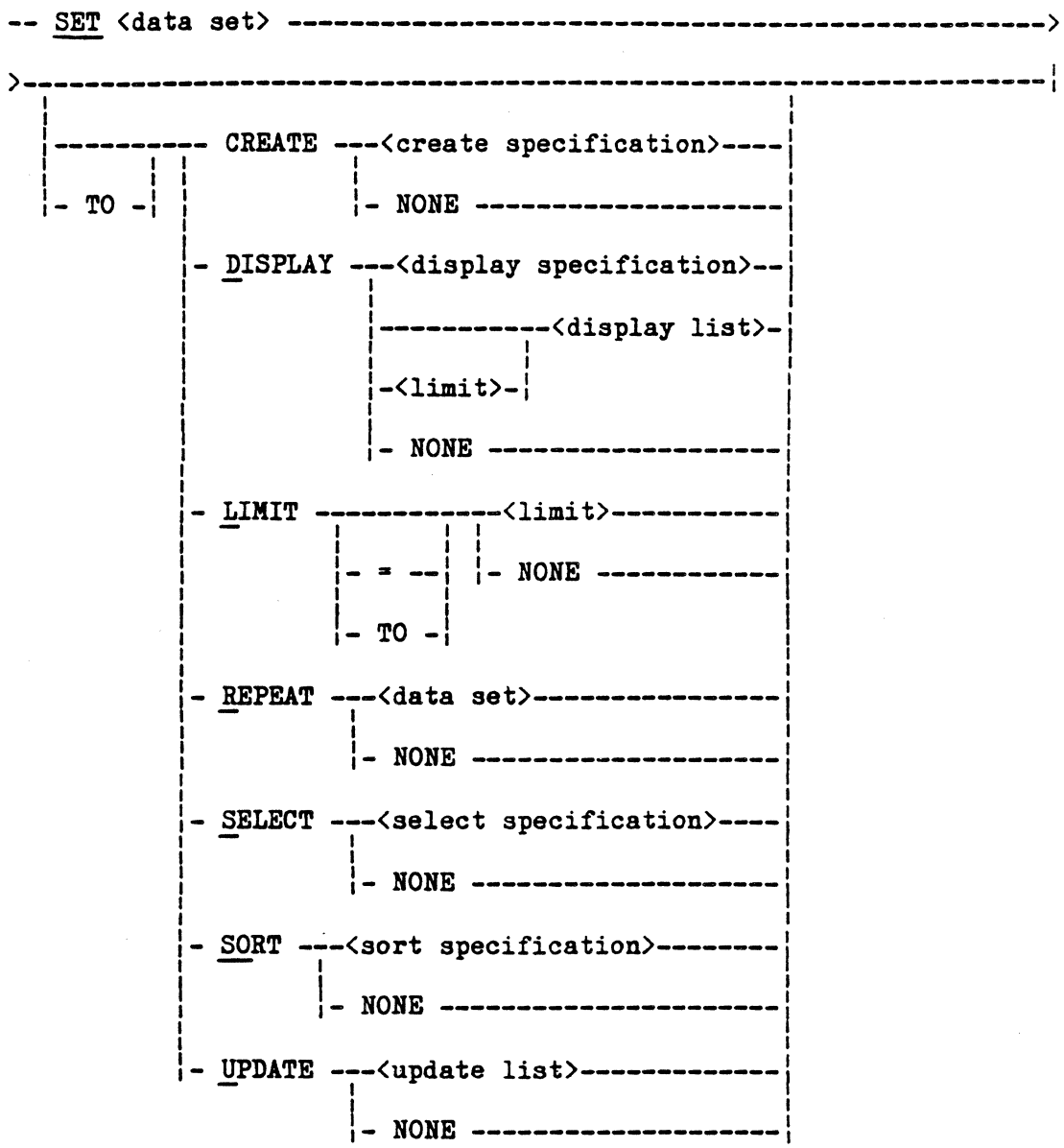
The SET command is used to associate the text of one or more Inquiry commands with a specified data set. In addition, Inquiry remembers the most recently entered text of CREATE, DISPLAY, LIMIT, SELECT, SORT, REPEAT, and UPDATE commands for each data set, and the SET command can be used to modify or delete the text of these commands stored in the specified data set. If new text is specified, the new text replaces the stored text; if NONE is specified, the stored text is deleted.

SET only changes the text associated with the data set; it does not select a record, update a record, or display an item.

The RECALL command may be used to load the text for commands associated with the data set into the current buffer. The EDIT command may be used to edit this text, and the edited command may then be executed using a REPEAT command or stored using a SET command.

The SET command may be used to establish a REPEAT command for a data set and, thus, automatic looping between that data set and an unrelated data set. (Refer to "REPEAT Command" in this section for an explanation of automatic looping. Examples of automatic looping may be found there or under "Related Records Across Data Sets" in Section 2.)

Syntax:



Semantics:

<data set>

Stores, modifies, or deletes command text associated with this data set.

TO
=

May be included to improve readability.

NONE

Eliminates the specified text.

CREATE

Changes or eliminates <create specification> associated with the <data set>.

DISPLAY

Changes or eliminates the <display specification> associated with the <data set>. A new <select specification> or <limit> can be included in the <display specification>. (Refer to DISPLAY Command for definitions of these constructs.)

LIMIT

Modifies or eliminates the <limit> associated with the <data set>. When LIMIT is set for a <data set>, Inquiry causes a page wait after <limit value> records have been displayed from the <data set>. To continue the display, transmit a blank.

LIMIT specifications are ignored for reports.

REPEAT

Modifies or eliminates the REPEAT specification associated with the <data set>.

SELECT

Modifies or eliminates the <select specification> associated with the <data set>. When the <select specification> is changed or eliminated, no current record remains selected for the <data set> or for any data set embedded within the <data set>.

SORT

Modifies or eliminates the <sort specification> associated with the <data set>.

UPDATE

Modifies or eliminates the <update list> associated with the <data set>.

Examples:

In the following example, the SET command is used to establish a DISPLAY command to be associated with ADR (which is then executed using a REPEAT command):

```
-> SET ADR TO DISPLAY 4 ADR AT ADR-EMP-NO < 8

#

-> REPEAT ADR

ADR-EMP-NO  ADR-STREET-NO                  ADR-ZIP
----------  -----------------------------  -------
         2  45 ALEXANDER RD DU BOIS PA       50007
         4  2345 HUNTINGTON ST READING PA    50007
         5  6 OLIVE AVE BELVEDERE NJ         56432
         6  23 PHILIPS ST APT 2 MARLENA GA   42323
 #
```

The following SET command changes the number of lines specified in the DISPLAY command associated with ADR:

```
-> SET ADR LIMIT TO 1

#

-> REPEAT ADR

ADR-EMP-NO  ADR-STREET-NO                  ADR-ZIP
----------  -----------------------------  -------
         2  45 ALEXANDER RD DU BOIS PA       50007
#
```

SHOW Command

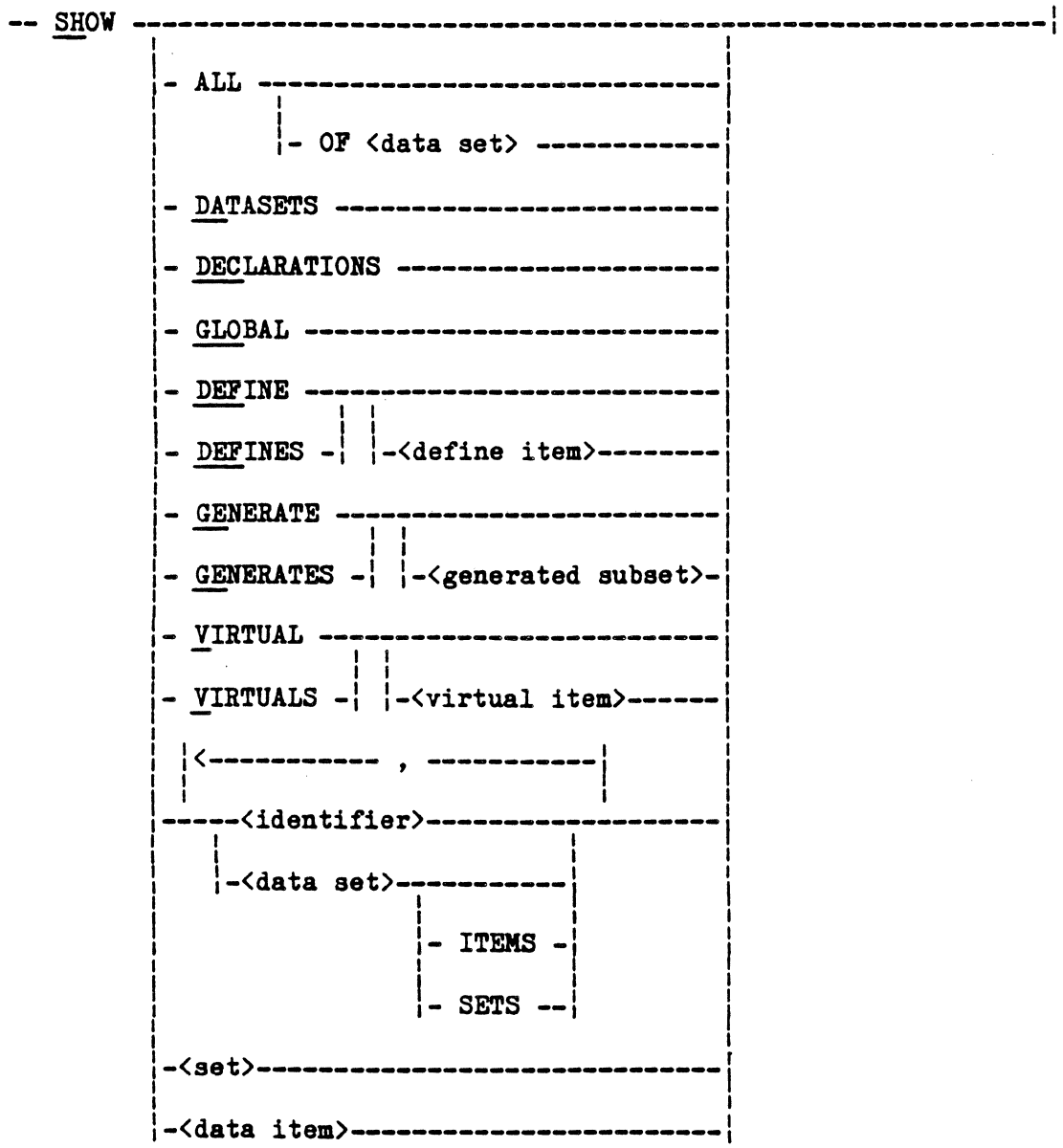
The SHOW command displays the contents of the current buffer, virtual items, define items, generated subset specifications, or specified parts of the DASDL declaration of a data base.

The ALL, DATASETS, DECLARATIONS, DEFINES, GENERATES, GLOBAL, and VIRTUALS options must be preceded by a pound sign (#) if the option name is also the name of a data base identifier. For example, if ALL is a data set name, SHOW ALL displays the description of the data set ALL, and SHOW #ALL displays a description of the entire data base.

If the COMMENTS option is TRUE (refer to OPTIONS Command in this section), comments appearing in the DASDL declaration of the data base are displayed.

SHOW output cannot be listed on the line printer.

Syntax:



Semantics:

SHOW

If no other options are specified, displays the contents of the current buffer.

ALL

Displays the DASDL description of the portion of the data base or logical data base that Inquiry was generated to access.

ALL OF <data set>
<data set>

Displays the DASDL description of the data items, sets, and subsets associated with this data set.

DATASETS

Displays the names of all data sets in the data base.

DECLARATIONS

Displays the names and text associated with all define items, generated subsets, and virtual items.

GLOBAL

Displays the description of all global items contained in the data base.

DEFINES
DEFINES
DEFINE <define item>

Displays the name and text for all define items or for the specified define item.

GENERATE
GENERATES
GENERATE <generated subset>

Displays the name and text for all generated subsets or for the specified generated subset.

VIRTUAL
VIRTUALS
VIRTUAL <virtual item>

Displays the name and text for all virtual items or for the specified virtual item.

<identifier>

Displays the DASDL description of the specified data set, set, or data item. If <identifier> is a data set identifier, this form is synonymous with SHOW ALL OF <data set>.

<data set> ITEMS
<data set> SETS

Displays the DASDL description of the data items in this data set or of the sets associated with this data set.

<set>

Displays the DASDL description of the set and indicates which data set the set references.

<data item>

Displays the DASDL description of the data item and the name of the data set that contains the item.

Examples:

In the following examples, SHOW commands are used to display the virtual item, generated subsets, and define item that have previously been declared for the sample data base:

```
-> SHOW VIRTUAL

   VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12
   #

-> SHOW GENERATES

   GENERATE RICHEMP = EMP-SALARY > 2000
   GENERATE OLDTIMERS = EMP-DATE-HIRED < 600101
   GENERATE TRUSTIES = RICHEMP AND OLDTIMERS
   GENERATE CRONIES = RICHEMP OR OLDTIMERS
   GENERATE PROTECTORS = RICHEMP + OLDTIMERS
   GENERATE DEFENDERS = RICHEMP - OLDTIMERS
   #

-> SHOW DEFINE

   DEFINE HIRERANGE (FDATE,LDATE) = D EMP AT EMP-DATE-HIRED > FDATE
    AND  EMP-DATE-HIRED < LDATE
```

```
#
```

In the following example, a SHOW command is used to display the DISPLAY command contained in the current buffer:

```
-> DISPLAY ADR AT ADR-EMP-NO < 4

   ADR-EMP-NO  ADR-STREET-NO                    ADR-ZIP
   ----------  -------------------------------  -------
            2  45 ALEXANDER RD DU BOIS PA         50007
   #

-> SHOW

   DISPLAY ADR AT ADR-EMP-NO < 4
   #
```

The following SHOW command displays the DASDL description of the global items in the sample data base:

```
-> SHOW GLOBAL

   GLOBAL ITEMS:
     FISCAL-YEAR NUMBER (6)
     POP-EMP POPULATION OF EMP
     DIV-NUMBER NUMBER (5)
   #
```

The following SHOW command displays the DASDL description of the item JOB-CODE in the JOB data set:

```
-> SHOW JOB-CODE

    JOB-CODE OF JOB NUMBER (4)
   #
```

The following SHOW command displays the DASDL description of the embedded set ED-DATE-SET:

```
-> SHOW ED-DATE-SET

    ED-DATE-SET OF EMP SET OF EMP-ED KEY ED-DATE
   #
```

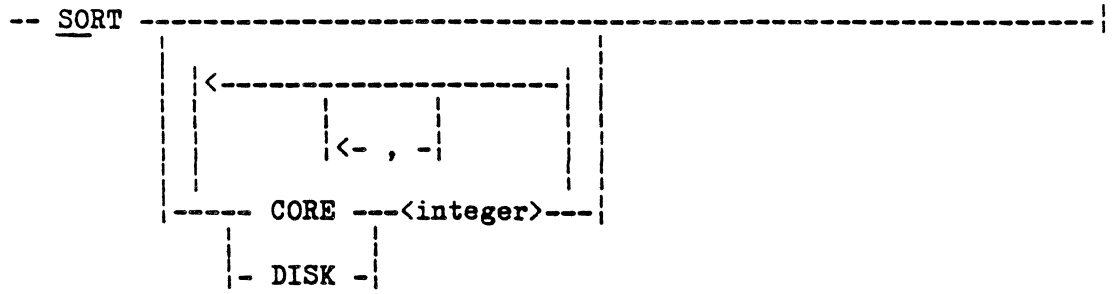
SORT Command

The SORT command displays or alters the SORT core and disk parameters used by the <sort option> of the SELECT and DISPLAY statements.

By default, the SORT is allocated 12,000 words of core and 600,000 segments of disk. In general, increasing the amount of core increases the speed of the sort.

The disk size specified must be sufficient to contain the tag file.

Syntax:



Semantics:

SORT

If no other options are specified, displays the current values for SORT core and SORT disk.

CORE <integer>
DISK <integer>

Assign new values for the SORT core and disk parameters.

Examples:

The following SORT command displays the current default values for the SORT core and disk parameters:

```
-> SORT

     CORE = 12000
     DISK = 600000
   #
```

The following SORT command changes the SORT core parameter to 13,000 words of core and the SORT disk parameter to 610,000 segments of disk:

```
-> SORT CORE 13000 DISK 610000

   #
```

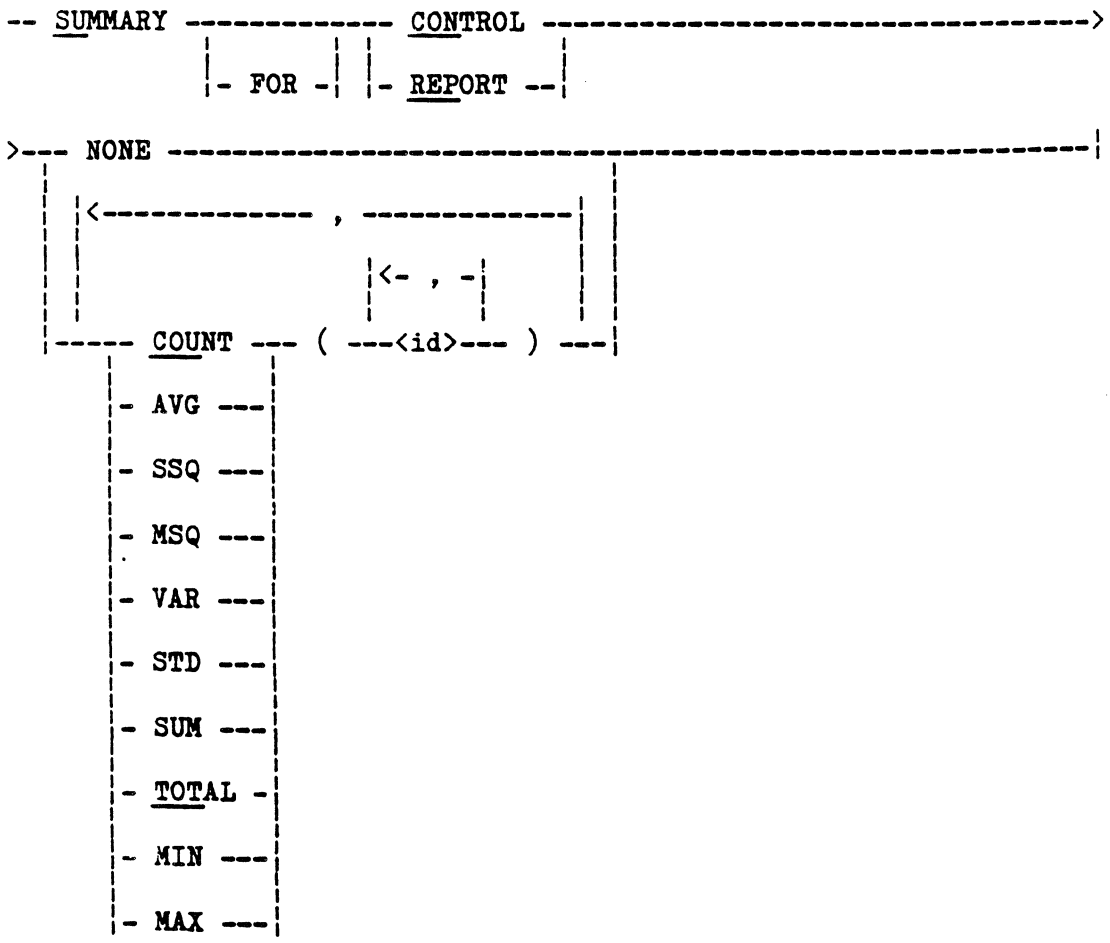
SUMMARY Command

The SUMMARY command allows statistical information to be included in a report. The functions used to produce the statistical information are defined under <arithmetic function> in "Common Syntactic Items."

Both SUMMARY REPORT and SUMMARY CONTROL can be used within a single report.

The REPORT command must be entered before entering a SUMMARY command.

Syntax:



Semantics:

FOR

May be included to improve readability.

CONTROL

Specifies that statistical information is to be processed and displayed at control breaks. A control break occurs each time a report control item in the report list changes value. (Refer to REPORT command for a definition of report control items.)

The order of control items is significant when more than one is listed. The first control item listed is the most important and the last is least important. When a control break occurs, control summaries for the item causing the break and items subordinate to it are printed. Summaries for the items superordinate to the item causing the break are not printed.

REPORT

Prints the report summary as an extra page after the report is generated.

NONE

Eliminates a previous SUMMARY command.

<id>

An <id> in the REPORT command <report list>. For SUMMARY CONTROL, <id> must be in the REPORT command <id list>. If a <colname> is specified for an item in the <report list>, <id> must be the <colname>. (Refer to REPORT Command in this section.)

TOTAL

Synonym for the arithmetic function SUM.

Examples:

In the following example, a number of employee records are selected from the data set EMP for a report. The SUMMARY command is then used to generate summary information for the sum of the salaries of the selected employees, the average of these salaries, the minimum salary, and the maximum salary along with the specified report information. This summary information is output as a separate page at the end of the report:

```
-> SELECT EMP AT EMP-NO < 5

-> REPORT EMP-LNAME,EMP-FNAME,EMP-SSN,EMP-DATE-HIRED,EMP-SALARY

   #

-> SUMMARY REPORT SUM(EMP-SALARY), AVG(EMP-SALARY), MIN(EMP-SALARY),%
   #%
-> MAX(EMP-SALARY)

   #

-> GENERATE REPORT VIA EMP

   #STARTING REPORT
                                                                    PAGE 1
        EMP-LNAME  EMP-FNAME    EMP-SSN EMP-DATE-HIRED EMP-SALARY
        ---------- ---------- --------- -------------- ----------
        FIELDS     JON        533733557          720312    5000.00
        HARDWICK   ALICE      647938292          701214    5000.00
        JONSON     BEN        540022677          620115    1771.56
        DEKKER     THOMAS     554998666          711104    1950.00

                                    .
                                    .
                                    .
                                                                    PAGE 2
  SUMMARY FOR REPORT

  EMP-SALARY :TOTAL=13721.56, AVG=3430.39, MIN=1771.56, MAX=5000

  #REPORT COMPLETED
  #
```

In the following example, the JOB-TYPE, JOB-CODE, and JOB-TITLE of every record in the JOB data set are output for a report. The SUMMARY command is then used to output the number of jobs that correspond to each JOB-TYPE along with the report information. This summary information is output at each control break (that is, each time the report control item JOB-TYPE changes value in the report).

```
-> REPORT JOB-TYPE:JOB-TYPE,JOB-TITLE,JOB-CODE

   #

-> SUMMARY CONTROL COUNT(JOB-TYPE)

   #

-> GENERATE REPORT VIA JOB

   #STARTING REPORT
```

182B

DMSII INQUIRY SOFTWARE OPERATION GUIDE

PCN 1164035-001 Release 3.6 December 1985

PAGE 1

```
JOB-TYPE : 0
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             0 VICE PRESIDENT                      9999
             0 ACCOUNTANT                          4000
             0 CONTROLLER                          8888
             0 ADM ASSIST                          7000
           ----
(SUBTOTAL
 FOR
JOB-TYPE)
COUNT        4

JOB-TYPE : 1
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             1 RECEPTIONIST                        2345
             1 SECRETARY                           1000
           ----
(SUBTOTAL
 FOR
JOB-TYPE)
COUNT        2
                .
                .
                .
```

PAGE 2

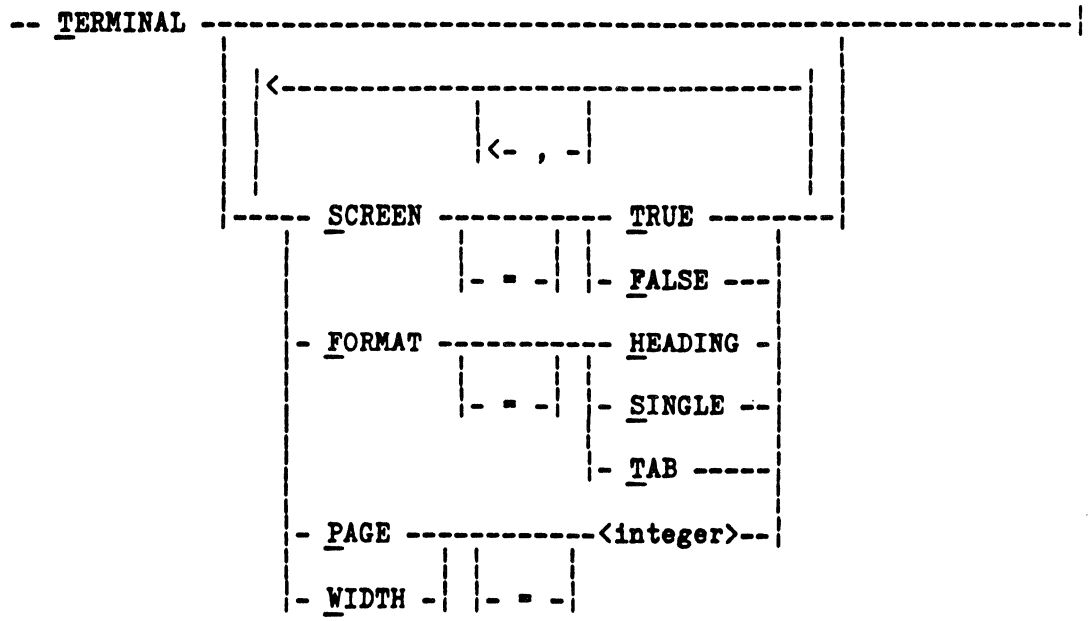
```
JOB-TYPE : 2
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             2 SALESMAN                            3333
             2 SALES MANAGER                       7777
             2 PROJECT LEADER                      5000
           ----
(SUBTOTAL
 FOR
JOB-TYPE)
COUNT        3

# REPORT COMPLETED
#
```

TERMINAL Command

The TERMINAL command displays or alters the values assigned by Inquiry for the terminal attributes SCREEN, FORMAT, PAGE, and WIDTH. The initial values for the PAGE, WIDTH, and SCREEN attributes depend on the terminal being used.

The TERMINAL command is ignored from card input.

Syntax:



Semantics:

TERMINAL

If no other options are specified, displays the current terminal attributes.

SCREEN

Must be TRUE for CRT terminals. SCREEN may be used with other terminal types to limit the amount of uninterrupted output.

PAGE 1

```
JOB-TYPE : 0
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             0 VICE PRESIDENT                     9999
             0 ACCOUNTANT                         4000
             0 CONTROLLER                         8888
             0 ADM ASSIST                         7000
           ====
COUNT        4

JOB-TYPE : 1
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             1 RECEPTIONIST                       2345
             1 SECRETARY                          1000
           ====
COUNT        2
                .
                .
                .
```

PAGE 2

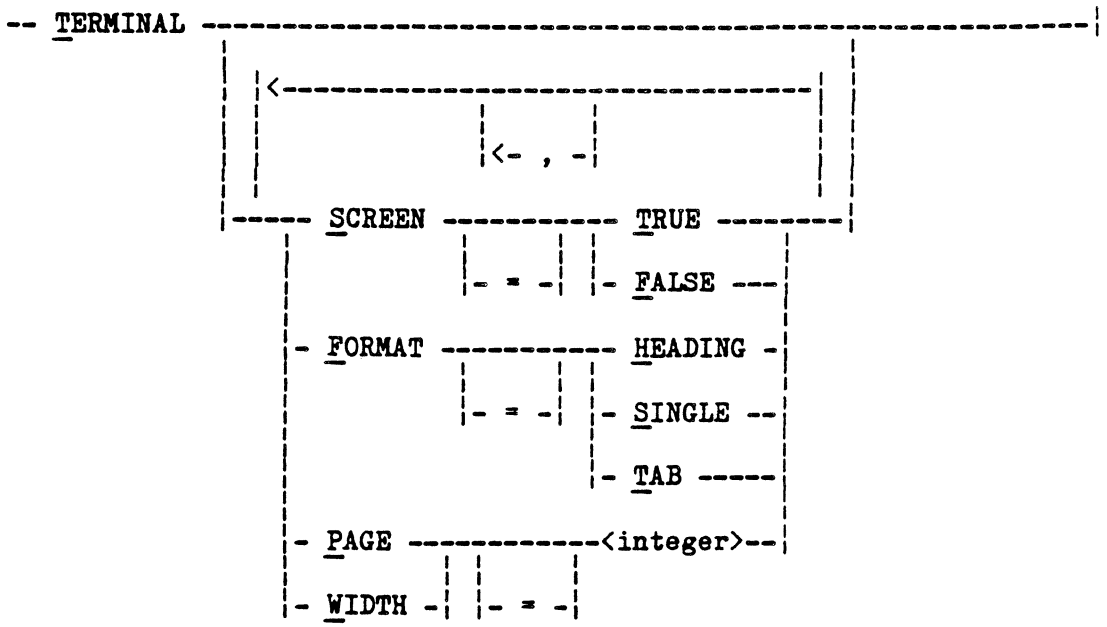
```
JOB-TYPE : 2
      JOB-TYPE JOB-TITLE                      JOB-CODE
      -------- ------------------------------ --------
             2 SALESMAN                           3333
             2 SALES MANAGER                      7777
             2 PROJECT LEADER                     5000
           ====
COUNT        3

# REPORT COMPLETED
#
```

TERMINAL Command

The TERMINAL command displays or alters the values assigned by Inquiry for the terminal attributes SCREEN, FORMAT, PAGE, and WIDTH. The initial values for the PAGE, WIDTH, and SCREEN attributes depend on the terminal being used.

The TERMINAL command is ignored from card input.

Syntax:



Semantics:

TERMINAL

If no other options are specified, displays the current terminal attributes.

SCREEN

Must be TRUE for CRT terminals. SCREEN may be used with other terminal types to limit the amount of uninterrupted output.

FORMAT

Specifies one of the three output formats.

HEADING
TAB
SINGLE

Refer to "Formatting Attributes" in Section 2.

PAGE

Specifies the number of lines displayed per page on CRT terminals.

WIDTH

Controls the number of characters displayed on each line. Some terminals are designed to perform an automatic line advance after printing in the last character position of the line; for this reason, Inquiry restricts output to one character less than the WIDTH value specified.

=

May be included to improve readability.

Examples:

The TERMINAL command in the following example displays the current values of the terminal attributes:

```
-> TERMINAL

    WIDTH = 70
    PAGE = 23
    SCREEN = TRUE
    FORMAT = HEADING
#
```

The TERMINAL command in the following example makes these specifications for terminal output:

1. In HEADING format.

2. 60 characters displayed per line.

3. 15 lines displayed per page on the terminal screen.

```
-> TERMINAL FORMAT=HEADING,WIDTH=60,PAGE=15

#
```

A DISPLAY command is then entered to show the results of these specifications:

```
-> DISPLAY JOB

1:JOB-COUNT  JOB-TYPE  JOB-CODE
2:JOB-TITLE                          TYPE-SPEED
3:SHORTHAND-SPEED  CRTUSE-REQD  SPANISH-REQD  DRIVING-REQD
4:JOB-COMMENT
------------------------------------------------------------
1:        1         0      9999
2:VICE PRESIDENT                          --
3:              --  --           --            --
4:--
1:        1         0      4000
2:ACCOUNTANT                              --
3:              --  --           --            --
4:--
```

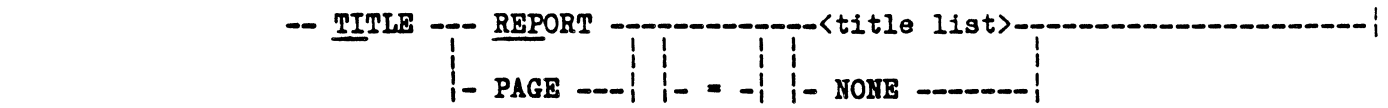
For additional examples, refer to Formatting Attributes in Section 2.

TITLE Command

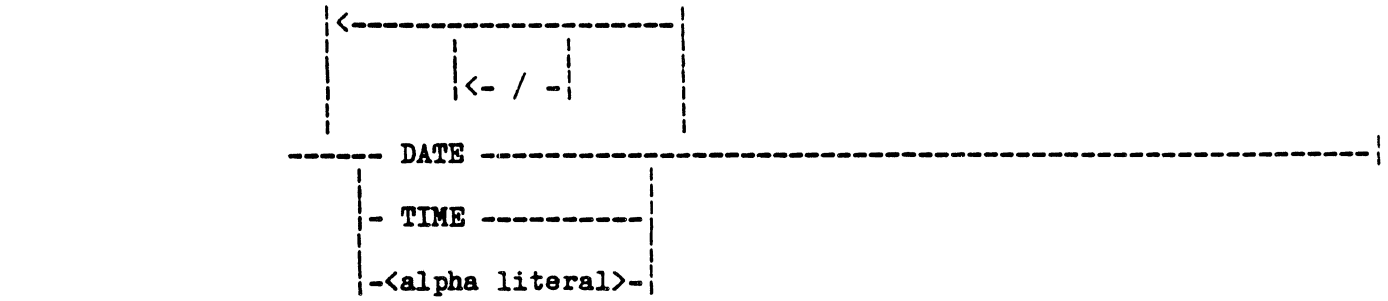
The TITLE command defines the specifications for report titles.

The length of a report-title line cannot exceed the line width of the output device. For reports directed to line printers, the line width is equal to the printer width. For reports directed to hardcopy terminals, the line width is equal to the terminal width. For reports directed to screen terminals, the line width is equal to the terminal width minus one column. In addition, the centered first line of a title appearing on every page of a report uses the last 10 columns for the page number; thus, the length of this line cannot exceed the line width minus 21.

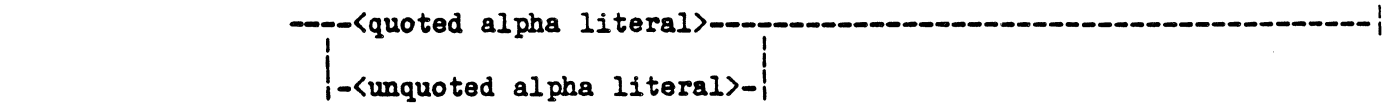
Syntax:



<title list>



<alpha literal>



Semantics:

REPORT

Defines a title page for a report. The title is centered on the page; however, a report title is optional.

PAGE

Defines a title for the top of every page of a report. This title is optional.

=

May be included to improve readability.

<title list>

Specifies contents of each report title line.

NONE

Eliminates a previous title.

DATE

Prints the current date as part of the title in the following form:

<month>/<day>/<year>

TIME

Prints the current time as part of the title in the following form:

<hour>:<minute>

<quoted alpha literal>
<unquoted alpha literal>

Defined in the semantics under <selection condition> in "Common Syntactic Items."

If OPTION QUOTES is TRUE, <alpha literal> must be a <quoted alpha literal>.

If OPTION QUOTES is FALSE, <alpha literal> can be either a <quoted alpha literal> or an <unquoted alpha literal>. However, the DATE and TIME options are not interpreted as <unquoted alpha literal>s, and any <alpha literal>s containing special characters or embedded blanks must be quoted.

Produces multi-line titles. Each occurrence of "/" indicates that the following title text appears on a new line.

Examples:

In the following example, the TITLE command is used to add a title page to the beginning of the report generated by the REPORT command in the example under REPORT Command in this section:

```
-> REPORT JOB-CODE:TITLE=JOB-TITLE AT 10 FOR 20,COMMENT=JOB-COMMENT %
   #%
-> AT 35 FOR 35

   #

-> TITLE REPORT = "MANAGEMENT JOB SUMMARY"/DATE/TIME

   #

-> GENERATE REPORT VIA JOB

   #STARTING REPORT


                          MANAGEMENT JOB SUMMARY
                                01/05/82
                                 14:15

                                   .
                                   .
                                   .

                                                                  PAGE 1

JOB-TITLE : PROJECT MANAGER
    CODE      COMMENT
    ----      ------------------------------------------------
    5000      3 YRS MANAGEMENT EXP PLUS DEGREE
```

```
JOB-TITLE : ADM ASSIST
    CODE      COMMENT
    ----      ------------------------------------------------
    7000      --


JOB-TITLE : SALES MANAGER
    CODE      COMMENT
    ----      ------------------------------------------------
    7777      5 YRS SALES EXP PLUS DEGREE

JOB-TITLE : CONTROLLER
    CODE      COMMENT
    ----      ------------------------------------------------
    8888      --
                                  .
                                  .
                                  .
                                                        PAGE 2
JOB-TITLE : VICE PRESIDENT
    CODE      COMMENT
    ----      ------------------------------------------------
    9999      --

# REPORT COMPLETED
```

The TITLE command in the following example places the title specified in the previous example at the top of each page in the report (new pages are indicated in this example by the vertical ellipses):

```
-> TITLE PAGE "MANAGEMENT JOB SUMMARY"/DATE/TIME

   #

-> GENERATE REPORT VIA JOB

   #STARTING REPORT

                        MANAGEMENT JOB SUMMARY
                              01/05/82
                              14:15

                                  .
                                  .
                                  .
```

```
                          MANAGEMENT JOB SUMMARY              PAGE 1
                                 01/05/82
                                  14:15
JOB-TITLE : PROJECT MANAGER
    CODE      COMMENT
    ----      ------------------------------------------------
    5000      3 YRS MANAGEMENT EXP PLUS DEGREE

JOB-TITLE : ADM ASSIST
    CODE      COMMENT
    ----      ------------------------------------------------
    7000      --

JOB-TITLE : SALES MANAGER
    CODE      COMMENT
    ----      ------------------------------------------------
    7777      5 YRS SALES EXP PLUS DEGREE


JOB-TITLE : CONTROLLER
    CODE      COMMENT
    ----      ------------------------------------------------
    8888      --
                                  .
                                  .
                                  .
                          MANAGEMENT JOB SUMMARY              PAGE 2
                                 01/05/82
                                  14:15

JOB-TITLE : VICE PRESIDENT
    CODE      COMMENT
    ----      ------------------------------------------------
    9999      --

#REPORT COMPLETED
#
```

For additional examples, refer to "Inquiry Report" in Section 2 and to "REPORT Command" in this section.

UPDATE Command

The UPDATE command modifies the value of data items in the currently selected record and then stores the updated record in the data set. Each data item specified in the UPDATE command is assigned the new value specified in the command.

All data items appearing in an UPDATE command must belong to the same data set. In addition, items used in an arithmetic expression that provides a value for an arithmetic item must be in the data set being updated, or in currently selected records of other data sets at a higher nesting level.

The following types of items cannot be updated:

1. Count items.
2. Key items for which no duplicates are allowed (the default condition for key items).
3. Link items.
4. Population items.
5. Record-type items.

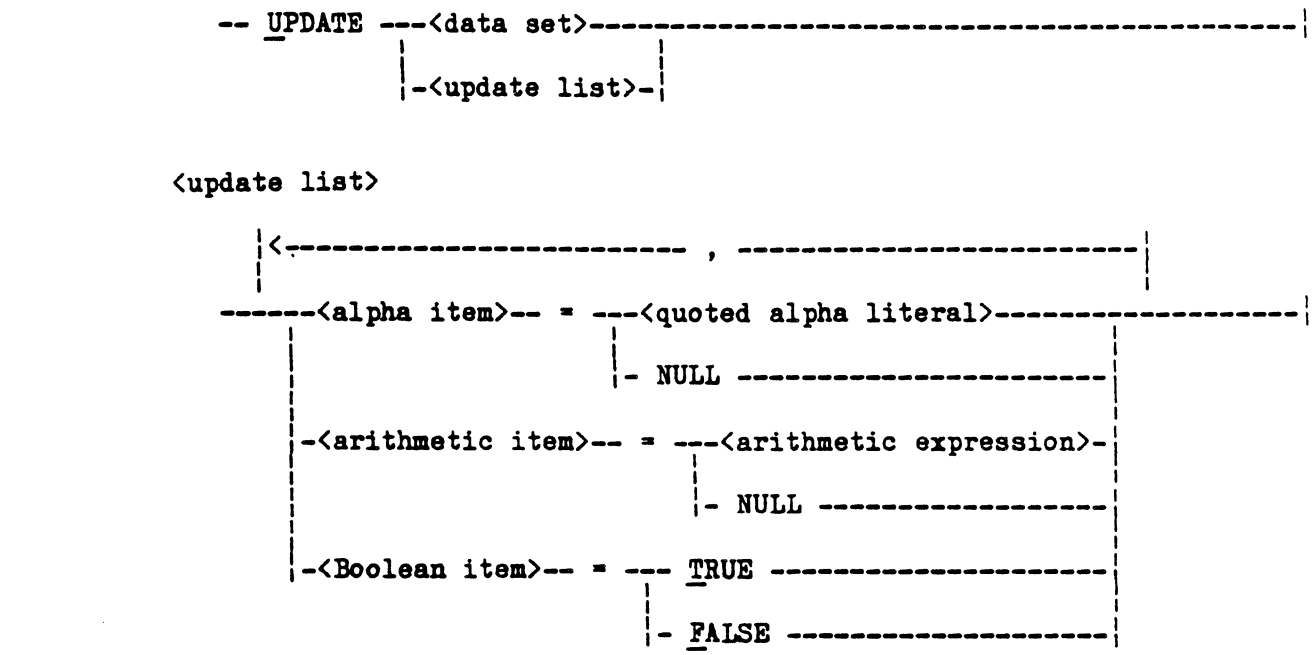
INQUIRY cannot update any embedded data set whose master (owner) data set has been compiled with the LOCK TO MODIFY DETAILS option set. If such an update is attempted, INQUIRY will respond with a "CANNOT MODIFY THIS EMBEDDED DATASET USING DMINQUIRY" message.

Inquiry remembers the most recent update list for each data set. This list can be accessed using a RECALL command or modified using a SET command. (Refer to "RECALL Command" and "SET Command" in this section.)

UPDATE invokes the data base ACCESSROUTINES. Any violations of the normal DMSII restrictions related to the STORE function cause an error condition and result in the display of an error message. If an error condition is detected, updating is not performed.

Updates made to an audited data base are audited in the usual DMSII manner. (Refer to "Modification of Audited Data Bases" in Section 2.)

Syntax:



Semantics:

<data set>

Executes the most current <update list> for this data set. If a record has not been selected, or an <update list> has not been specified for this data set, an error message is displayed and no update is performed.

<update list>

Specifies items to be updated in the current record of the selected data set and the new values for these items.

<alpha item>
<arithmetic item>
<Boolean item>

Items in the currently selected record.

<quoted alpha literal>

Defined in the semantics under <selection condition> in "Common Syntactic Items." Must be enclosed in quotes even if OPTIONS QUOTES = FALSE. (Refer to "OPTIONS Command" in this section.)

<arithmetic expression>

Defined under <selection condition> in "Common Syntactic Items."

NULL

Sets data items to their DASDL-defined null values. Required items and items such as Boolean items that cannot take null values cannot be set to NULL.

Examples:

In the following example, the first record of the data set EMP is selected, and the items EMP-SALARY and EMP-SEC-CLEAR are updated. The updated record is then displayed:

```
-> SELECT EMP

   #

-> UPDATE EMP-SALARY = 5000, EMP-SEC-CLEAR = TRUE

   SELECTED RECORD OF EMP UPDATED
   #

-> D ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------
   1:      1  FIELDS      JON         533733557          720312
   2:TRUE              5000.00
   #
```

The next record of EMP is then selected, updated, and displayed. (Because the same update list is used to update the second record, only the name of the data set is specified in the UPDATE command.)

```
-> NEXT
```

```
-> UPDATE EMP

   SELECTED RECORD OF EMP UPDATED
   #

-> DISPLAY ALL

   1:EMP-NO  EMP-LNAME   EMP-FNAME     EMP-SSN  EMP-DATE-HIRED
   2:EMP-SEC-CLEAR  EMP-SALARY
   ------------------------------------------------------------------
   1:     2  KRANTZ      ALICE       647938292          701214
   2:TRUE              5000.00
   #
```

The following example shows how a record from an embedded data set is updated. First a record is selected from both the master data set EMP, and the associated record in the embedded data set EMP-ED is selected. Then, the UPDATE command is executed for the selected embedded record:

```
-> SELECT EMP

   #

-> SELECT VIA ED-DATE-SET

   #

-> DISPLAY ALL

   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 650615  KENT STATE                            2123  PSYCHOLOGY
   2:  3.77
   #

-> UPDATE ED-INST-NAME = "CARLETON ",ED-INST-CODE = 7174

   SELECTED RECORD OF EMP-ED UPDATED
   #

-> DISPLAY ALL

   1:ED-DATE  ED-INST-NAME                    ED-INST-CODE  ED-COURSE
   2:ED-GPA
   ------------------------------------------------------------------
   1: 650615  CARLETON                              7174  PSYCHOLOGY
   2:  3.77
   #
```

For additional examples, refer to "Update" in Section 2.

VIRTUAL Command

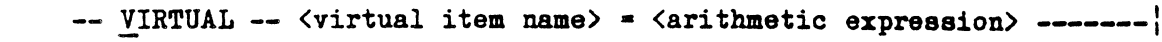
The VIRTUAL command creates a virtual item and associates it with the specified arithmetic expression.

Virtual items can be used in display lists or selection conditions. All Inquiry rules concerning the use of arithmetic expressions and functions must be followed.

If the arithmetic expression contains functions, these functions must apply to items contained in records at the next lower nesting level. (Refer to Functions in Section 2.)

Inquiry recognizes when the value of a virtual item must be recomputed. When this situation occurs, Inquiry recomputes the arithmetic expression and assigns the new value to the virtual item.

Syntax:



Semantics:

<virtual item name>

A unique identifier.

<arithmetic expression>

Defined under <selection condition> in "Common Syntactic Items."

Examples:

In the following example, the virtual item EMP-YEAR-SALARY is created based on the item EMP-SALARY:

```
-> VIRTUAL EMP-YEAR-SALARY = EMP-SALARY * 12

   #
```

```
-> DISPLAY EMP-LNAME, EMP-SALARY, EMP-YEAR-SALARY AT EMP-NO > 5

   EMP-LNAME   EMP-SALARY     EMP-YEAR-SALARY
   ----------  ----------  ------------------
   MARLOWE        2200.00                26400
   BURNS          1750.00                21000
   MITTY          2000.00                24000
   JOYCE          2450.00                29400
   STACK          4500.00                54000
   #
```

For additional examples, refer to "Virtual Items" in Section 2.

APPENDIX A. HOW TO READ RAILROAD DIAGRAMS

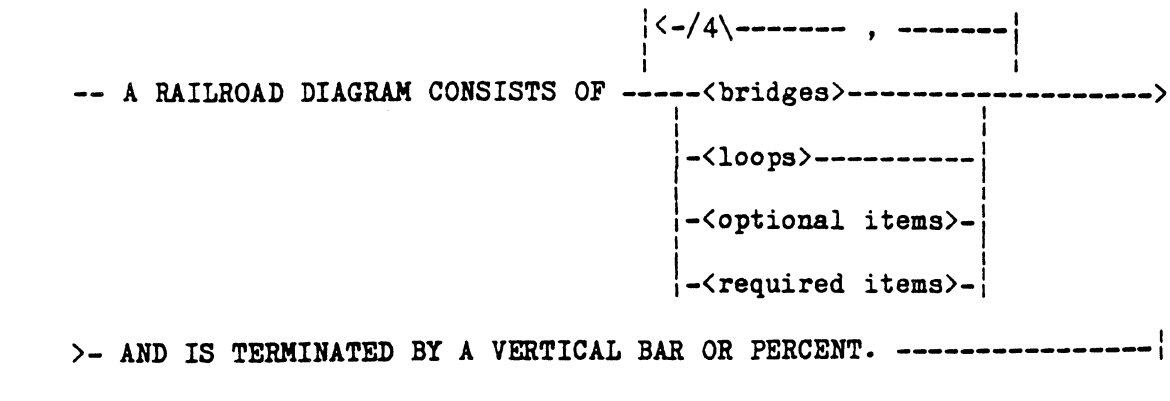
Railroad diagrams are graphic representations of the syntax.

The railroad diagrams are traversed left to right or in the direction of the arrowhead. Adherence to the limits illustrated by bridges produces a syntactically valid statement. Continuation from one line of a diagram to another is represented by a right arrow (>) appearing at the end of the current line and the beginning of the next line. The complete syntax diagram is terminated by a vertical bar (!) or a percent sign (%).

Items contained in angle brackets (<>) are syntactic variables that are defined in the manual or are information that the user is required to supply.

Uppercase items not enclosed in angle brackets must appear literally; the minimum abbreviations are underlined.

Example:



The following are some of the syntactically valid constructs that may be generated from the above diagram:

A RAILROAD DIAGRAM CONSISTS OF <bridges> AND IS TERMINATED BY A VERTICAL BAR OR PERCENT.

A RAILROAD DIAGRAM CONSISTS OF <optional items> AND IS TERMINATED BY A VERTICAL BAR OR PERCENT.

A RAILROAD DIAGRAM CONSISTS OF <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR OR PERCENT.

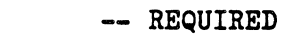
A RAILROAD DIAGRAM CONSISTS OF <optional items>, <required items>, <optional items>, <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR OR PERCENT.

Railroad Components

<required items>

No alternate path through the railroad diagram exists for required items or required punctuation.

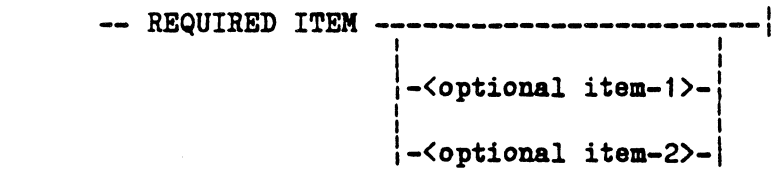
Example:



<optional items>

Items shown as a vertical list indicate that the user may make a choice of the items specified. An empty path through the list allows the optional item to be absent.

Example:



The following valid constructs may be generated from the above diagram:

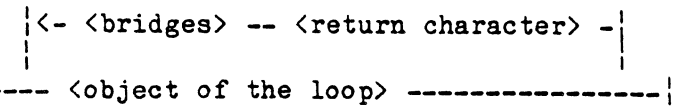
REQUIRED ITEM

REQUIRED ITEM <optional item-1>

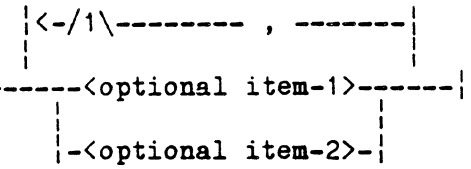
REQUIRED ITEM <optional item-2>

<loops>

A loop is a recurrent path through a railroad diagram and has the following general format:



Example:



The following are some of the valid constructs that may be generated from the above diagram:

<optional item-1>

<optional item-1>,<optional item-1>

<optional item-2>,<optional item-1>

The loop must be traversed in the direction in which the arrowhead points, and the limits specified by bridges cannot be exceeded.

<bridges>

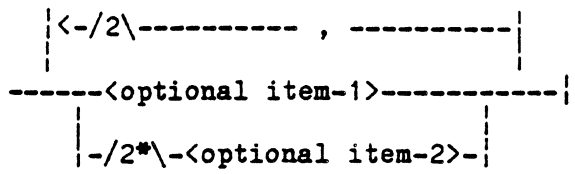
A bridge illustrates the minimum or maximum number of times a path may be traversed in a railroad diagram.

Three forms of bridges exist:

/n\ n is an integer that specifies the maximum number of times the path may be traversed.

/n*\ n is an integer that specifies the maximum number of times the path may be traversed. The asterisk (*) indicates that the path must be traversed at least once.

Example:



The loop may be traversed a maximum of two times, and the path for <optional item-2> must be traversed at least once but no more than twice.

The following are some of the valid constructs that may be generated from the above diagram:

<optional item-1>,<optional item-2>

<optional item-2>,<optional item-2>,<optional item-1>

<optional item-2>

B GLOSSARY

The following section contains a glossary of terms used in this manual.

ACCESSROUTINES

Routines that perform all physical and logical management of a data base and allow many users to access the data base concurrently. Each data management statement that a user language program executes invokes a portion of the ACCESSROUTINES to perform all file management functions that the statement requires.

alpha item

A data item that contains character information (letters, numbers, special characters, and blanks) as EBCDIC characters. An alpha item cannot be used in calculations.

alpha literal

Refer to "quoted alpha literal" and "unquoted alpha literal" in this glossary.

arithmetic function

A function containing calculations that produce a numeric result based on one or more records. (Refer to "current record function" and "data set function" in this glossary.)

audited data base	A data base that stores a record of changes (called the "audit trail"), which can be used for data base recovery in the event of hardware or software failure.
automatic looping	The Inquiry process in which the commands associated with one data set are executed every time a REPEAT command associated with a second data set, and specifying the first data set, is executed.
Boolean item	A data item that stores information having a logical value of either TRUE or FALSE.
calculator mode	The condition in which Inquiry is used to display the results of an arithmetic expression involving numeric constants but not data base items.
CANDE	See "Command AND Edit."
Command AND Edit (CANDE)	A Message Control System (MCS) that provides generalized file preparation and updating capabilities and task control in an interactive, terminal-oriented environment.

comments	Descriptive information included in data base description tables for a data item.
control item	A count item, population item, or record-type item.
count item	A control item that contains a system-maintained count of the number of counted links that refer to the record containing it. The use of a count item allows read-only access to the count field of a record. Any data set referenced by a counted link must contain a count item. Records that contain a nonzero count cannot be deleted.
current buffer	A buffer established in Inquiry to record the most recently entered DEFINE, DISPLAY, GENERATE, SELECT, SET, or VIRTUAL command.
current record	The most recently selected record in a given data set.

current record function	An arithmetic function performed on a specified data item in the currently selected record. Current record functions in Inquiry are ABS, ARCCOS, ARCSIN, ARCTAN, COS, EXP, LN, LOG, SCALE, SIN, SQRT, and TAN.
DASDL	See "Data and Structure Definition Language."
Data and Structure Definition Language (DASDL)	The language used to describe a data base logically and physically, and to specify criteria to ensure the integrity of data stored in the data base. DASDL is the source language that is input to the DASDL compiler, which creates or updates the data base description file from the input.
data base	An integrated, centralized system of data files and program utilities designed to support an application. The data sets and associated index structures are defined by a single description. Ideally, all the permanent data pertinent to a particular application resides in a single data base.
data item	A field in a data base record or transaction format that contains a particular type of information. For example, EMP-LNAME is a data item in the sample data base. (Refer to "alpha item," "number item," "Boolean item," and "group item" in this glossary.)
data set	A collection of related data records stored in a file on a random-access storage

	device. A data set has physical properties similar to those of a file; however, a data set may contain other data sets, sets, or subsets.
data set function	An arithmetic function that produces summary information from a number of data set records. Inquiry data set functions are AVG, COUNT,MAX, MIN, MSQ, SSQ, STD, SUM, and VAR.
declarations	A general term used to refer to the texts of DEFINE, GENERATE, and VIRTUAL commands as a group.
define item	An identifier that is used as a shorthand notation for a usually longer piece of Inquiry text. This text can be an entire command, several commands, or a piece of a command.
definitions file	A collection of declarations established and saved during previous Inquiry sessions.
disjoint data set	A data set that is not embedded in another data set.
DMALGOL	Data Management ALGOL.
DMSII	Data Management System II.
embedded data set	A data set that is an item within another data set. Records of an embedded data set must be accessed through the "master" data set in which it is embedded.
embedded structure	An embedded set, data set, or subset.
extract file	A data file that is produced by extracting information from a data base using the Inquiry EXTRACT command. The format of the file is specified in the EXTRACT command.

field item	A data base item that can store unsigned integer or Boolean values.
generated subset	A temporary subset, based on frequently used selection conditions, established using a GENERATE command.
global item	An item declared in DASDL that generally contains control information pertaining to an entire data base. For example, POP-EMP, the item containing the number of items in the data set EMP, is a global data item in the sample data base.
group item	A data item made up of a collection of smaller data items.
Inquiry session	The period between the time Inquiry is initiated and the time Inquiry is terminated.
item	A field in a data base record that contains an individual piece of information and can be referenced by name.
link item	A data base item that allows one data set record to refer to another.
logical data base	A collection of structures declared in DASDL that can be used to provide a "view" of the data base, enforce structure-level security, and achieve data independence. Programs that invoke a logical data base cannot access structures that were not included in the logical data base definition.
logical operator	One of the operators AND, OR, and NOT used to specify logical operations in Inquiry selection conditions.
manual subset	A collection of indices or pointers to records in a single data set. A manual subset is used to access selected members

	of that data set and to represent relationships between data records. Entries in a manual subset must be maintained programmatically, not by DMSII, because manual subsets do not specify WHERE conditions.
master data set	A data set containing an embedded data set.
null	The value contained in an item when that item does not contain valid information.
number items	A data item that consists of a number that can be used in calculations.
OCCURS clause	Clause in a DASDL data item declaration that establishes an ordered (subscripted) collection of data items with identical attributes.
parametric define	A define item that includes a variable item (the "parameter") whose value must be supplied every time the define item is used.
population item	A control item that contains the number of records in a given data set. (Refer to "control item" in this glossary.)
qualification	Specification of the data set that owns an item. Qualification is usually used when several data sets contain an item with the same name.
quoted alpha literal	A quoted sequence of uppercase characters A-Z, lowercase characters a-z special characters, and blanks.
record	A collection of data items that are usually related in some logical way (For example, an entry for an employee in the EMP data set is a record.)

record-type item	A control item that contains the format number for records in variable-format data sets. (Refer to "control item" in this glossary.)
record layout summary	A summary of an extract file, containing the names, types and offsets of extracted items, produced when the extract file is created.
report control item	A data item specified in a REPORT command that groups records with the same value for that item on the report.
selection condition	The part of an Inquiry statement that determines the condition a record must meet to be included in a Inquiry operation (for instance, a DISPLAY or a SELECT command).
set	A collection of information that DMSII uses to locate records in a data set. In Inquiry, sets can be used in SELECT or DISPLAY commands to designate how a data set should be accessed.
subset	A collection of indices or pointers to the records of a single data set A subset is used to access selected members of that data set and to represent relationships between the data set records.
swapspace	A collection of "chunks" of memory used for job swapping. (Refer to Chapter 16 of the A Series System Software Operational Guide, Volume 2, for a description of job swapping.)
sync-point	A point in time when no program updating the data base is in a transaction state, and a sync-point record is written to the data base. (Refer to "transaction" and "transaction state" in this glossary.)
tag file	File created during an Inquiry sort that contains key items and addresses of

	selected records and is passed to the system SORT intrinsic.
transaction	A sequence of data-management operations grouped by a user program as a single operation to ensure that these operations are executed as a single change to a data base. (In other words, either all transaction changes are made or none are made.)
transaction state	The period in a user program between a BEGINTRANSACTION and an ENDTRANSACTION.
unquoted alpha literal	A sequence of characters containing uppercase characters A-Z, lowercase characters a-z, and numeric characters 0-9. An unquoted alpha literal may not contain special characters or embedded blanks.
variable-format data set	A data set whose records may have several different formats. All records in such a data set, known as "variable-format records," consist of a common "fixed part" and optionally include one of the variable parts declared in the DASDL.
virtual item	In Inquiry, an item established using the VIRTUAL command that represents frequently used arithmetic expressions. (This definition of "virtual item," not the DASDL definition, is that used throughout this manual.)
WFL	Work Flow Language.
ZIP	An ALGOL statement that causes the WFL compiler to be initiated. It is commonly used to automatically initiate compilations.

C GENERATING AND RUNNING INQUIRY

Inquiry can be generated to access an entire data base or selected data sets within the data base. Inquiry can be generated for a maximum of 700 structures.

Modification of the data base may be allowed or disallowed when Inquiry is generated.

Multiple copies of Inquiry may be generated for a data base. Each copy of Inquiry may access different data sets, and each may allow or disallow data base modification.

INQUIRY FILES

Inquiry is generated from the following files:

DMINQ/BUILDSYMBOLIC	Symbolic of the program that is compiled to produce SYSTEM/BUILDINQ. DMINQ/BUILDSYMBOLIC can be compiled using ALGOL by equating ALGOL file TAPE to the file DMINQ/BUILDSYMBOLIC.
SYSTEM/BUILDINQ	Object code file for DMINQ/BUILDSYMBOLIC. SYSTEM/BUILDINQ builds a directory of the portion of the data base that Inquiry is able to access. This directory is included in the Inquiry code file for run-time interpretation of the data base structure.
DATABASE/DMINQ	Symbolic file for Inquiry.
DMINQ/HELP	A file containing the information returned to the user in response to the HELP <verb>, HELP VERB, and HELP VERBS forms of the HELP command. If this file is not present, HELP INFORMATION NOT AVAILABLE is displayed when one of these forms of the HELP command is entered.

INQUIRY GENERATION

The following paragraphs describe the required operations for generating Inquiry from remote terminals and card readers.

Generation From Remote Terminals

Inquiry is generated from remote terminals through CANDE using the following procedure:

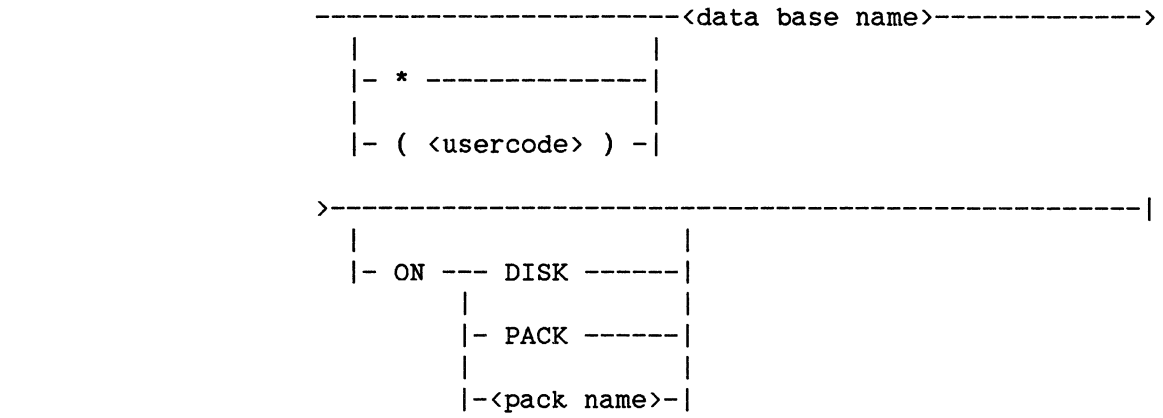
1. The user runs the BUILDINQ program by entering the following statement:

   ```
   RUN $SYSTEM/BUILDINQ
   ```

2. BUILDINQ responds with the following display message:

   ```
   WHAT DATABASE?
   ```

   The user then enters the data base name. The syntax for the data base name is:



   The data base name specification must appear in uppercase letters; if not, BUILDINQ displays the following message and then terminates:

   ```
   ***** ERROR *****  BAD DATABASE NAME SYNTAX
   ```

If BUILDINQ cannot locate the file DESCRIPTION/<data base name>, or if the specified data base name is incorrect, BUILDINQ displays the following message and then terminates:

```
****** ERROR *****  NO SUCH DATA BASE
```

3. If the DESCRIPTION/<data base name> file and the DATABASE/PROPERTIES file are both present, BUILDINQ displays the following message:

```
WHICH SOFTWARE?
1 DMINQUIRY
2 DMINTERPRETER
```

The user enters 1 to specify that BUILDINQ is to build a directory for the Inquiry program.

4. BUILDINQ then displays the following message:

```
ALLOW Inquiry ONLY (YES OR NO)?
```

The user enters YES to disallow the UPDATE, CREATE, and DELETE commands or NO to allow any of these statements. If NO is entered, BUILDINQ displays the following message:

```
ALLOW UPDATE (YES OR NO)?
```

A YES response to this prompt allows UPDATE; a NO response disallows UPDATE. BUILDINQ then asks:

```
ALLOW CREATE (YES OR NO)?
```

YES allows CREATE; NO disallows CREATE. BUILDINQ then asks:

```
ALLOW DELETE (YES OR NO)?
```

YES allows DELETE; NO disallows DELETE.

5. BUILDINQ then asks:

```
WHICH OPTION

1 TOTAL DATABASE
2 SELECTED DATASETS
3 LOGICAL DATABASE
```

If no logical data bases are declared for the data base, option 3 is not displayed. The user enters 1, 2, or 3 to specify which portion of the data base is to be accessible to Inquiry.

Option 1 allows Inquiry access to all data base structures.

Option 2 allows Inquiry access to selected data base structures. If this option is chosen, BUILDINQ displays the following prompt:

```
FOR FOLLOWING DATASETS ENTER

   E (OR EMPTY) TO EXCLUDE FROM SOFTWARE
   I TO INCLUDE INTO SOFTWARE
   C TO INDICATE DONE
   * TO TERMINATE
```

BUILDINQ then displays the name of each disjoint data set in the data base and waits for a response after each name is displayed. The user must enter E, I, C, or * to specify the action to be performed for each displayed data set. If * is entered, BUILDINQ is terminated and must be restarted. If C is entered, BUILDINQ stops displaying the list of data sets and proceeds to step 6.

Option 3 allows Inquiry access to one logical data base. If this option is chosen, BUILDINQ displays the following prompt:

```
ENTER LOGICAL DATABASE NAME
```

At this point, the user must enter the name of the logical data base. The name entered for the logical data base must not specify a logical data base in which an embedded set is invoked more than once, since this would result in a "REMAP EMBEDDED SET INVOKED TWICE" error.

6. When the desired option has been selected, BUILDINQ asks:

```
INQUIRY PROGRAM NAME (NULL FOR DEFAULT)?
```

"(NULL FOR DEFAULT)" appears in the display only if option 1 or option 3 has been selected and the data base name does not contain any special characters. If a null input is transmitted in response to the prompt, the following default name is used for the Inquiry program:

```
OBJECT/INQUIRY/<inquiry name>
```

In this form, <inquiry name> is the data base name if option 1 was selected or the logical data base name if option 3 was selected.

If the user wishes to supply a name for the Inquiry program, or if option 2 was selected, the Inquiry program name has the same form as the default name, except that <inquiry name> is the name supplied by the user. The supplied name may contain from 1 to 11 identifiers separated by slashes. Each identifier may contain from 1 to 17 alphanumeric characters. Care must be taken in selecting an Inquiry program name because any existing

file with the same name as the specified <inquiry name> is removed.

7. When the Inquiry program has been named, BUILDINQ asks:

WHAT QUEUE (NULL FOR DEFAULT)?

If a null input is entered, Inquiry is compiled in the system default queue; otherwise, Inquiry is compiled in the queue specified by the user.

8. When the queue has been selected, BUILDINQ asks:

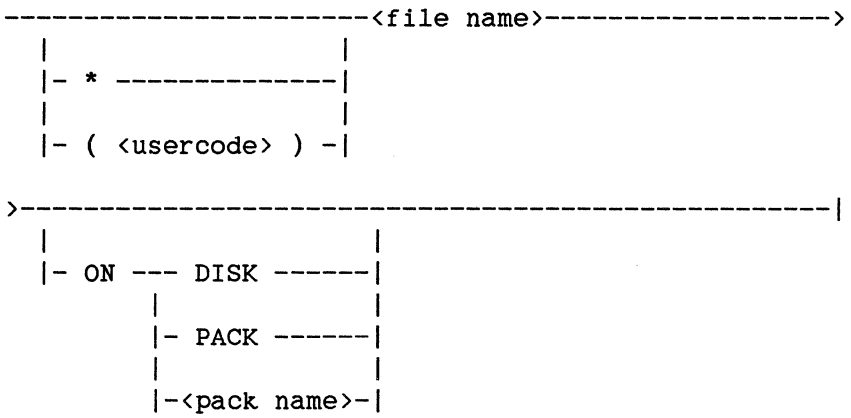
DEFINITION FILE NAME(NULL FOR DEFAULT)?

The user can then assign a name to the default definitions file used by the Inquiry SAVE and RESTORE commands. If a null input is entered, the default definitions file has the following form:

(<usercode>)DEFINITIONS/<inquiry name>

In this form, <usercode> is the Inquiry user's usercode, not the usercode under which BUILDINQ is run.

If the default name is not desired, the definitions file name may be specified using the following syntax:



In this form, <file name> may contain from 1 to 11 identifiers separated by slashes. Each identifier may contain from 1 to 17 alphanumeric characters. If a usercode is specified, nonprivileged users running under other usercodes are not able to access the definitions file.

When the default definitions file has been named, BUILDINQ performs an ALGOL ZIP to compile the Inquiry program. The result of the ZIPed compilation is a code file with the following title:

OBJECT/Inquiry/<inquiry name>

This title may be prefixed by a <usercode>. The code file is placed on the device specified by the <usercode> and FAMILY statement.

Remote Generation Using The Nozip Option

Normally, BUILDINQ produces the Inquiry directory with the title DMINQDIRECTORY/<inquiry name> and then performs a ZIP to compile Inquiry. However, the ZIP can be inhibited using the $NOZIP option. The user may then compile Inquiry using a job deck rather than allowing BUILDINQ to ZIP the compilation. The $NOZIP option is used as described in the following paragraphs.

1. The Inquiry remote-generation procedure proceeds as normal; the user enters the appropriate CANDE RUN command for BUILDINQ, and BUILDINQ displays the request:

   WHAT DATABASE?

   At this point, instead of entering the data base name, the user enters:

   $NOZIP

2. BUILDINQ accepts this entry and then repeats the original prompt:

   WHAT DATABASE?

   The data base name should then be entered. When $NOZIP is used, BUILDINQ does not request the queue specification. If BUILDINQ is run from cards, the QUEUE statement is ignored, if present.

To compile the Inquiry program after using the $NOZIP option, the following cards must be included in a job deck and executed:

```
  COMPILE OBJECT/Inquiry/<inquiry name> WITH DMALGOL LIBRARY
  COMPILER FILE TAPE = DATABASE/DMINQ
  COMPILER FILE DASDL = DMINQDIRECTORY/<inquiry name>
  COMPILER FILE PROPERTIES = DATABASE/PROPERTIES
? COMPILER DATA
$ MERGE
? END
```

When compiled in this way, instead of OBJECT/Inquiry/<inquiry name> the name of the Inquiry facility can be specified to be any name the user desires.

The file-equation cards may require modification, depending on the location of the files being used.

Generation From Job Decks

BUILDINQ can also be run using a job deck. The following cards may appear in the job deck, in any order:

DATABASE : <data base name>

If this card is present, the data base description file is found in a file titled DESCRIPTION/<data base name>. If this card is omitted, the user must provide the title of the data base description file by label-equating the DASDL file of BUILDINQ.

$ NOZIP

If this card is present, BUILDINQ produces the DMINQDIRECTORY but does not compile Inquiry.

UPDATE

If this card is present, the data base can be updated using Inquiry.

CREATE

If this card is present, records can be added to the data base using Inquiry.

DELETE

If this card is present, records can be deleted from the data base using INQUIRY.

INQNAME : <inquiry name>

If this card is present, the generated Inquiry program is named (<usercode>)OBJECT/INQUIRY/<inquiry name>; otherwise, the default name (<usercode>)OBJECT/INQUIRY/<data base name> is used, unless the deck contains an LDBNAME card. In that case, the default name (<usercode>)OBJECT/INQUIRY/<logical data base name> is used.

LDBNAME : <logical data base name>

If this card is present, access to the specified logical data base is assumed; otherwise, access to the total data base is assumed. <logical data base name> must refer to a logical data base declared in DASDL. If the logical data base invokes the same structure more than once, the error message "STRUCTURE INVOKED TWICE" will be displayed and the generation halted.

QUEUE : <integer>

If this card is present, INQUIRY is compiled in the specified queue; otherwise, INQUIRY is compiled in the system default queue.

DEFNAME : <definitions file name>

If this card is present, the definitions file is named (<usercode>)DEFINITIONS/<definitions file name>; otherwise, the default name (<usercode>)DEFINITIONS/<data base name> is used.

The CARD input file for BUILDINQ may be label-equated to a file of type DISK, REMOTE, or READER.

If an error is encountered when the BUILDINQ job deck is run, an appropriate error message is displayed and BUILDINQ terminates.

Examples:

The following job deck uses the DATABASE card to specify the data base whose description file is used to build INQUIRY:

```
?BEGIN JOB BUILDINQFROMCARDS;
RUN SYSTEM/BUILDINQ;
DATA
DATABASE : MYDB
QUEUE : 40
DEFNAME : MY/DEFINITIONS
?END JOB.
```

The following job deck label-equates the DASDL file title for the data base whose description file is used to build INQUIRY:

```
?BEGIN JOB BUILDINQFROMCARDS;
RUN SYSTEM/BUILDINQ;
FILE DASDL (TITLE=DESCRIPTION/MYDB);
DATA
QUEUE : 40
DEFNAME : MY/DEFINITIONS
?END JOB
```

INITIATING AND RUNNING INQUIRY

The following paragraphs describe the operations required to initiate and run Inquiry once the program has been successfully compiled.

Required Files

All data base files plus the following files must be present to run Inquiry:

SYSTEM/ACCESSROUTINES	File containing the data base ACCESSROUTINES (routines that perform all physical and logical management of a data base).
OBJECT/Inquiry/<inquiry name>	This file is the code file from the compiled Inquiry program.

The user may initiate Inquiry through CANDE by entering the following statement:

```
RUN Inquiry/<inquiry name>
```

The user may also initiate Inquiry using a job deck that includes the following cards:

```
RUN OBJECT/Inquiry/<inquiry name>
FILE REMOTE (TITLE = <station name>)
```

Efficiency Considerations

Inquiry can be run either in main memory or in swapspace.

If one terminal inquires against the data base at a time, Inquiry should be run entirely in swapspace (SUBSPACE=3).

If several users are running Inquiry concurrently, Inquiry should be run with code in main memory and data in swapspace (SUBSPACE=1). In this case, a single copy of the code can be shared by all users.

If only a few (three or four) terminals inquire against a data base but many batch inquiries occur, Inquiry should be run entirely in main memory (SUBSPACE=0).

For other combinations of batch and remote users, Inquiry should be run entirely in swapspace (SUBSPACE=3).

Run-time Diagnostic Options

By default, Inquiry is compiled with the ARRAYS, BASE, FAULT, and CODE task options SET. Thus, any fault-termination of Inquiry produces a program dump.

In addition, if Inquiry input produces error messages, additional diagnostic information can be obtained using Inquiry diagnostic options. To use these options, the user enters the following two statements:

```
$ LINENUM
$ DUMP
```

The user should then repeat the Inquiry command that caused the error. The $DUMP option causes a program dump to occur when the error is encountered again. The $LINENUM option causes Inquiry to display the line number in DATABASE/DMINQ at which the error occurred.

These options are RESET by entering the following statements:

```
$ - LINENUM
$ - DUMP
```

Operational Description

Inquiry may be run either from remote terminals or from card readers. The following paragraphs describe the operations for each kind of run.

Running Inquiry From Terminals

Inquiry is run from remote terminals using CANDE. After the CANDE log-on procedure has been performed, the user initiates Inquiry by entering the following command:

```
RUN Inquiry/<inquiry name>
```

<inquiry name> is the name of the data base or the name selected during Inquiry generation. If the user is not allowed access to the data base, Inquiry responds:

```
#INITIALIZING Inquiry
#ACCESS DENIED
```

If the user is allowed to access the data base, Inquiry responds:

```
#INITIALIZING Inquiry
#<data base name> READY
```

At this point, the user can enter Inquiry statements.

Normally, Inquiry commands are entered on a single line and executed immediately. However, an Inquiry command may exceed the line width of the terminal; in this case, a percent sign (%) may be entered at the end of the line to indicate that more input is follow. When the line is transmitted, Inquiry responds with a pound sign followed by a percent sign (#%) to indicate that additional input may be entered. This process can be repeated as many times as necessary. When the command is complete, the user can transmit the last line; the entire Inquiry command is then executed. A percent sign should not be entered for the last line of the command.

The user may wish to enter an input command but delay the processing of this statement until a later time. To cause Inquiry to store such a command, the user enters the command as described above; however, the last line of input must end with two percent signs (%%). The user then uses the REPEAT command to execute the command. The SHOW command may be used to display the stored command before it is executed; if desired, the EDIT command may then be used to modify the stored command. (Refer to EDIT Command, REPEAT Command, and SHOW Command in Section 3.)

Several Inquiry commands can be entered for immediate execution using a single, multiple-command input. A multiple-command input is created by using semicolons (;) as separators between the commands entered as input.

Some Inquiry input commands may produce a large amount of output. In these cases, the user can discontinue output by pressing the terminal BREAK key. When BREAK is entered, Inquiry discards all remaining

output, displays a pound sign (#), and waits for new input.

Some Inquiry statements may require a long time to produce an answer. If Inquiry detects this condition, it displays a #WAIT message.

The user can interrupt output at any time by using the BREAK key, or by entering the CANDE ?BRK command if the terminal has no BREAK key. When output is interrupted, Inquiry discontinues the current command, displays #STOPPED, and waits for new input.

NOTE

A pound sign (#), or any system-produced message preceded by a pound sign (except #WAIT), is an indication that Inquiry is waiting for additional input from the user.

Running Inquiry From Card Readers

Inquiry is also able to accept card input. When a card deck is punched to run Inquiry, the deck must specify a file CARD (with KIND=READER or label-equated to another medium). The deck should contain unsequenced Inquiry commands formatted in the same way as those entered from terminals. When Inquiry is run, the CARD file is opened (instead of the REMOTE file), and the associated file attributes are examined. OPTIONS is automatically set to PRINTER.

The output from an Inquiry batch run contains the input messages printed along with the associated output messages. No output is directed to the REMOTE file.

ABS Function, 19
ARCCOS Function, 19
ARCSIN Function, 19
ARCTAN Function, 19
<arithmetic function>, 68
Arithmetic Functions, 18
Attach Command, 91
Audited Data Bases
   Modification of, 31
AVG Function, 18

Basic Inquiry Process, 7
<boolean function>, 66
Boolean Functions, 17

Card Readers
   Running Inquiry From, 224
Clear Command, 93
Combining Record Selection And Item Display, 13
Commands, 49
   Summary of, 49
Common Syntactic Items, 53
COS Function, 19
COUNT Function, 18
Create And Delete, 28
Create Command, 97
Current Record Functions, 73

Data base
   Sample, 5
Data Base Modification, 26
Data Bases
   Modification of Audited, 31
Data Set Functions, 69
Data Sets
   and Related Records, 24
Define Command, 101
Delete and Create, 28
Delete Command, 104
Detach Command, 106
Diagnostic Options, 222
Display
   of Item and Value, 12
   Select, 88
Display Command, 32, 80
<display list>, 59

Edit Command, 108
Embedded Structures, 15, 33

EXP Function, 19
Extract
   Inquiry, 46
Extract Command, 112

File Command, 121
Files, 212
   Required, 221
Formatting, 32
Formatting Attributes, 35
Functions, 66
   Arithmetic, 18
   Boolean, 17
   Data Set, 69
   Record, 73
   Use of, 17

Generate Command, 125
Generating And Running Inquiry, 211
Generating Inquiry, 213
   from Remote Terminals, 213
Generation From Job Decks, 218
Glossary, 203

Heading, 35
Help Command, 132

Initiating And Running Inquiry, 221
Inquiry Commands, 49, 76
   Summary of, 49
Inquiry Extract, 46
Inquiry Files, 212
Inquiry Generation, 213
Inquiry Process, 7
Inquiry Report, 41
Item And Value Display, 12
Item Display
   Combining with Record Selection, 13
Items
   Undefined, 23
   Virtual, 21

Job Decks
   Generation From, 218

LN Function, 19
LOG Function, 19

MAX Function, 18
Mean Square Function, 18
MIN Function, 18
Modification Of Audited Data Bases, 31
Modifying Data Bases, 26
MSQ Function, 18

Natural Log Function, 19
Next Command, 134
Nozip Option
   Remote Generation Using, 217

Options Command, 136
Output Control And Formatting, 32

Printer Command, 140

Qualification, 11
Quit Command, 144

Railroad Components, 200
Railroad Diagrams, 199
Recall Command, 145
Record Functions, 73
Record Selection, 8
   Combining with Item Display, 13
Redefine Command, 148
Related Records Across Data Sets, 24
Remote Generation Using The Nozip Option, 217
Repeat Command, 150
Report Command, 153
Required Files, 221
Restore Command, 163
Run-time Diagnostic Options, 222
Running and Generating Inquiry, 211
Running and Initiating Inquiry, 221
Running Inquiry From Card Readers, 224
Running Inquiry From Terminals, 222

Sample Data base, 5
Save Command, 166
SCALE Function, 19
Select Command, 76
Select/Display Command, 88
Selection
   Record, 8
<selection condition>, 53

Selection Conditions, 9
Set Command, 169
Show Command, 173
Single, 38
Sort Command, 178
<sort option>, 62
SQRT Function, 19
Square Root Function, 19
SSQ Function, 18
Standard Deviation Function, 18
STD Function, 18
Structure Designation, 10
SUM Function, 18
Sum Of Squares Function, 18
Summary Command, 180
Summary Of Inquiry Commands, 49
Syntactic Items, 53

Tab, 37
TAN Function, 19
Terminal Command, 184
Terminals
   Running Inquiry From, 222
Title Command, 187

Undefined Items, 23
Update, 26
Update Command, 192
Use Of Functions, 17

Value and Item Display, 12
VAR Function, 18
Virtual Command, 197
Virtual Items, 21

116403500P002